THE FINAL TECHNICAL REPORT

for

THE DEVELOPMENT OF A FEATURE-BASED CONCURRENT ENGINEERING SYSTEM

(Grant # MDA 972-92-J-1008)

submitted to

Defense Advanced Research Project Agency
DARPA/DSO
3701 North Fairfax Drive
Arlington, Virginia 22203-1714
Attn: Dr. H. Lee Buchanan III

by

Dr. Chin-Sheng Chen
Industrial & Systems Engineering Department
Florida International University
University Park, ECS 416
Miami, Florida 33199
Tel: (305) 348-3753

October 10, 1994

Final Technical Report

Period: January 1, 1992 - August 31, 1994
Grant No.: MDA 972-92-J-1008

Title: **Development of a Feature-Based Concurrent Engineering System**

Principle Investigator: Dr. Chin-Sheng Chen/Dr. Fred W. Swift
Organization: Florida International University

Date: October 10, 1994

SUMMARY

This research project is aimed at development of a concurrent engineering system for integrated product and process development of machined component designs. Its main purpose is to evaluate the feasibility of implementation of a concurrent engineering system which provides both horizontal and vertical integration of CAD/CAM systems. The research and development of this project has been founded on concepts of form features and object-oriented computer techniques. The R&D tasks identified in the research project are: 1) feature definition and classification, 2) product data representation, 3) product information modeling, 4) manufacturing resources modeling, 5) producibility evaluation, 6) manufacturing process planning, and 7) NC program modeling. Throughout the entire project period, industrial practice and standards have been closely observed and incorporated in the development, when applicable.

The concurrent engineering system, coded in C++, runs on a Unix OS system. The workstations on which the system was tested are SUN/SPARC 10 and DEC 5000/200. The third party software packages required to run the system are the VERSANT object-oriented database management system and the ACIS geometric modeler. The database system is used to store product and its derived models, manufacturing resources data, and manufacturing planning knowledge. The geometric modeler is used to execute geometric computation at request of the product/feature modeler. The concurrence of the following engineering activities has been tested: product modeling, producibility evaluation, process planning, and NC programming.

This research project has attracted numerous inquiries from the industry as well as the academia. Research results have been presented to many companies such as Pratt and Whitney, Ford Motors, Cognition, Standard Technology, and the NCMS. Research findings have been published in a journal paper and several conference proceedings. Three additional journal papers are currently in review. Four more journal papers are in preparation. The research effort still continues to transfer the research results to the industry, universities, and other agencies

TABLE OF CONTENTS

1. SYSTEM OVERVIEW

The concurrent engineering system is configured in Figure 1. Broadly this system consists of two components: engineering data and engineering applications. The engineering data is composed of several data models required to store product and process data created by an engineering application and to support system-wide applications. Engineering data of particular interest to this project are: 1) manufacturing resources, 2) product definition, 3) producibility, 4) process plans, and 5) NC programs. Manufacturing knowledge is also viewed as a part of the engineering data. A representation scheme has been developed for each of the above data types, after an in-depth examination of their potential applications and modeling techniques. During the entire project period, these representation schemes were examined and modified constantly as engineering applications were applied to these models and needs for improvement were identified.

A CSG/Brep hybrid and feature-centered product data representation scheme has been developed to capture product designs. It is considered innovative with its ability to consolidate life cycle product data in one model, capture both feature volume and surface, retain modeling history, and support feature editing and modification. For process plans, an object-oriented representation scheme has been developed to formalize the feature-centered process plan. A classification and definition method for manufacturing resource data has been developed to uniquely identify, capture, and test attributes required to support multiple engineering applications. Manufacturing knowledge is classified and organized according to (manufacturing) form feature, machine, and tool (F/M/T) types. In correspondence to each F/M/T combination, a manufacturing knowledge module is constructed for mainly process planning. All engineering data models are stored in the VERSANT object-oriented database management system. A manufacturing resource manager has been developed to facilitate the interface between the user and the system. It allows the user to browse, add, delete, modify, and search the database. Separately, a knowledge manager has also been created for a similar purpose. It allows the user to modify, add, or delete manufacturing knowledge to/from the database system.

The engineering applications implemented in this system are: 1) product modeling, 2) producibility evaluation, 3) process planning, and 4) NC programming. The product modeler adopts the CSG input method and generates product models according to the proposed feature-

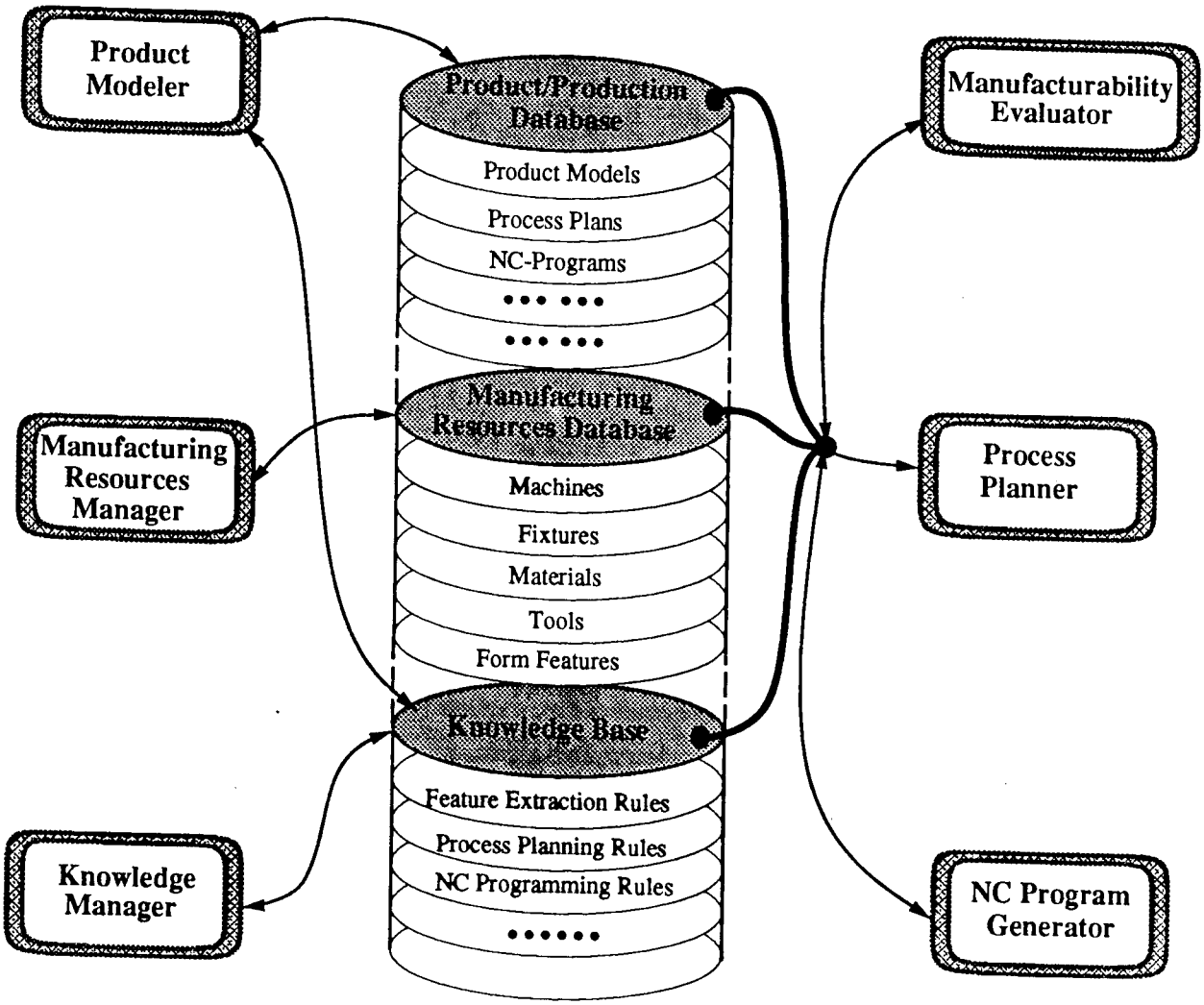


Figure 1: Concurrent Engineering System Architecture

based product data representation scheme. It uses the ACIS geometric modeler to perform geometric computation. The producibility evaluator evaluates a product information model in two steps. In the first step, a technical feasibility check is performed on a pass/fail basis. If it is passed, a producibility index is assessed for the design in the second step.

The process planner converts design features from a product model into manufacturing features for machining operations and applies manufacturing knowledge stored in the database to select a machine and cutting tool(s) for each manufacturing feature. After a sufficient number of alternative process plans are evaluated, the best plan is selected according to a given criterion and saved in the database system according to the proposed process plan representation scheme. The NC programmer retrieves a process plan for manufacturing feature, machine, and tool data, converts them into required tool paths, and generate an NC program for a given machine controller.

The entire system has been coded in C++ and runs on a Unix OS system. Currently it requires the support of two commercial software packages: the ACIS geometric modeler and the VERSANT object-oriented database management system. The system has been tested on SUN/SPARC 10 and DEC 5000/200 stations with multiple users on the network at remote sites. A number of product designs including the ANC-101M test part have been used to walk through the system and to evaluate its support for creating an integrated and concurrent product and process development environment. In each walk-through test, the product modeler was activated to model a product design. The purpose was to evaluate the efficiency of the proposed feature-based product modeling approach, and its flexibility and data consistency in feature editing and product modification. After a design was completed, the producibility evaluator was applied to the product model to test effectiveness of the feature-based product model in support of producibility evaluation.

The process planner was used to interpret a product design both before and after a product design passed the producibility evaluation to measure the combined validity (and efficiency) of the knowledge base, the manufacturing resource data, and the product model in support for process planning. Modularization of manufacturing knowledge is a direct result of the efficiency evaluation. The NC program modeler was applied to evaluate the validity of the proposed process plan model to support automatic NC programming and consequently the validity of the

proposed product model. The evaluation helps in improving the definition of manufacturing features and island constraints in the process plan model and including more lower-level geometric data in the product model. With these tests and improvement, now the system runs consistently and satisfactorily for machined part designs with up to 2 and 1/2 D form features and for manufacturing systems with up to 3-axis machine tools.

The rest of this report is organized into 10 sections. The next section is a description of the classification and information definition of form features. The third section is a description of the classification and definition of manufacturing resource data. The four following sections focus on a description of the four engineering applications implemented in this project. They are product modeling, process planning, producibility evaluation, and NC programming, respectively. The implementation of the system is presented in section 8, followed by a report on industrial collaboration in section 9 and conclusion in section 10.

2. FORM FEATURE DEFINITION AND CLASSIFICAITON

Based on the conventional CSG input process adopted in this project, form features are defined as a function of the base part (S_1), the creating solid primitive (S_2), the Boolean operator (B), the relative orientation of the creating solid (O_r), and the relative location of the creating solid (L_r). A form feature (FF) thus can be symbolically expressed as:

$$FF \approx f\ (S_1, S_2, B, O_r, L_r).$$

The base part (S_1) is the first creating solid primitive in the modeling process. During the modeling process, the base part may change in geometry, topology, and dimensions due to addition of other form features, but the first creating solid usually prescribes the overall shape of the part design. The other creating solid primitives (S_2) are also selected from the same set of system-defined primitive solids and add to the current design through a Boolean operator. These primitives typically have the shape of a cylinder, cuboid, wedge, prism, torus, and cone. Other shapes may be added later to customize the modeling system. The shape of each creating solid primitive prescribes the overall shape of the form feature. The Boolean operator (B) selected to perform the geometric computation determines the overall topology of the form

feature being created. A valid union operation creates a protrusion feature, while a subtraction creates a depression. The location of the creating solid with respect to the face on which the feature is located is used to signify a possible modification to the topology of the form feature. The orientation of a form feature is defined by the orientation of the creating solid with respect to the face on which it is created. It is considered in the feature definition because it captures some implication of machining difficulty.

The name of a form feature is important to human interpretation and thus the development of various manufacturing applications. Both the feature classification and the feature naming mechanism have been developed according to the above five factors. First of all, all features created on a cuboid base part are categorized as prismatic form features in this study. Within the product domain, at the first tier of feature classification, form features are labeled as prismatic features, cylindrical features, sheet metal features, and so on, depending on the product domain of interest. Within the domain of prismatic parts, for example, a form feature is further classified and named according to the remaining four factors. In this implementation, each feature name is structured with the following four segments of information: 1) the modifier of S_2, 2) the general shape of S_2, 3) the relative orientation of S_2 on S_1, and 4) Boolean operation & relative location of S_2 on S_1. The S_2 modifier specifies a modification to the face relationship on the wall faces of a form feature. It is defined as either regular or rounded. The S_2 general shape is defined as the shape of the creating solid, which can be cuboid, cylinder, cone, N-sided prism, and wedge, etc. The relative orientation is either 1D, 2D, or 3D. The Boolean operation and relative location in combination define a segment of a feature name and can be a boss if the operator is union or a depression if it is a subtraction. In the case of a valid subtraction, the feature name is further classified as through_step, blind_step, through_slot, blind_slot, through_hole, or pocket. A pocket is defaulted to a blind_hole, if the S2 is a cylinder.

In the domain of prismatic part designs, if a valid Boolean subtraction operation is performed to subtract a cylinder vertically from the top face of a cuboid base part, the form feature just being created is called a *regular-cylindrical-1D-blind_hole*. If the S_2 were a cuboid with a rounded corner radius on each corner between the wall faces on the cuboid, the form feature would be named as *rounded-cuboid-1D-pocket*. The feature naming strategy has been developed with general use in mind. The support for machining process planning was also a

focus in developing the naming mechanism. However, its applicability to other manufacturing processes such as casting and extrusion should be further evaluated.

Similar to the naming policy, the base part was used as the top tier identifier to classify form features. The creating primitive (S2) was used as the second tier classifier. The feature topology (its relative location & the Boolean operator) was used as the third tier classifier. The S2 modifier and the relative orientation were then applied as the last-tier classifiers. Secondary form features such as threads and knurls are not defined in this project. Compound features and complex features are not explicitly captured either. The feature hierarchy and feature illustrations are presented in Appendix A.

3.MANUFACTURING RESOURCE DATA

The modeling of manufacturing resources data in this project is limited to general-purpose machines, tools, fixtures, and material stocks which are used in a typical machine shop. Machines are limited to 3-axis NC and non-NC machines and are classified into 7 general and 17 specific machine classes. The general classes are mill, turn, thermal, saw, drill, EDM, and grind. A rich set of machine attributes are identified and attached to machines at various class levels. Machine attributes typically describe machine capability, tool compatibility, and fixture compatibility. Cutting tools are classified at tool assembly level, assuming cutting tools on the shop floor are assembled and stored in a tool crib or magazine ready for retrieval. A total of 24 tool classes were identified and structured to support the above machine types. To support milling machines, for example, milling tools are classified into four tool types: end mill, peripheral mill, slot mill, and face mill. A tool assembly structure has also been prepared for each assembly type to configure new tool assemblies in case that a feasible tool assembly is not available. The construction of tool assemblies, however, is not yet implemented. If it were implemented, the system would first search for a desired tool assembly in existence. When no such a tool assembly existed, the system would then activate the construction program to build a new tool assembly for the application. When a new tool assembly is designed, its assembly instruction would be also generated and sent to the tool room to physically assemble this new tool.

The fixtures for holding the workpiece a machine table are limited to machine vises. No

modular and custom-made fixtures are modeled in this study. A fixture class structure was developed to specify the compatibility among these fixture components in order to reflect their full applicability in practice. Material stocks are defined in terms of their machining related attributes and classified into sub-classes according to its sectional shape in order to capture its dimensions and to facilitate stock selection. Typical shape classes including square, rectangular, pentagonal, hexagonal, octagonal, and cylindrical are defined. The definition and classification of these manufacturing resources are presented in Appendix B.

4. PRODUCT MODELER

In a typical feature modeling system, a product is captured with a feature graph in which each node is a form feature and each arc links a pair of adjacent features. Existing feature modeling systems have been developed with emphasis on their efficiency in retrieving and interrogating feature information. With respect to feature recognition and feature editing support, however, feature shape information and feature modeling history have to be extracted directly from a geometric model of the underlying CAD system. This situation results in difficulty in both extracting form features unambiguously and editing a product model at the feature level. In this project we developed a CSG/Brep hybrid product data representation scheme and an innovative product/feature modeling system which captures both feature shape information and feature modeling history, while retaining feature adjacency relationships.

This product modeling system contains a product modeler, a feature editor, a feature extractor, a dimensioning/tolerancing modeler, and feature rule manager. The product modeler invokes the ACIS to create a product geometric model and to generate its corresponding feature model. An input to the product modeler is a CSG tree. During geometric modeling, the CSG tree is evaluated into a B-rep form by the ACIS geometric modeler. At the same time, the feature shape description related to the primitive solid being used in the Boolean operation is attached to this B-rep model and a feature graph is also constructed. Form features may be recognized at the modeling stage using the feature shape description on the B-rep. The feature extractor matches a piece of feature shape information with feature patterns in the feature rule library. It assigns a feature name to this piece of feature shape description. If the feature cannot be matched with all available feature rules in the feature library, the extractor will organize the

feature shape description into a new feature rule and will prompt the designer for a name. This new rule is then added to the feature library by the feature rule manager for later use. In other words, the feature rule library grows as models are added. A new feature rule can be input manually using the feature rule manager. The feature rule manager is responsible for resolving possible conflicts and inconsistencies among feature rules in the feature library.

The feature editor provides a flexible way to modify a product at the feature level. The feature editor is different from a conventional geometric editor in that it not only carries out geometric modification of a product design but also handles the changes of its features, feature graph, and associated manufacturing attributes. To efficiently attach manufacturing attributes such as datums, dimensional tolerances, and geometric tolerances to all geometric levels, the dimensioning/tolerancing modeler allows designer to click topological elements from a product on the screen and attach desired manufacturing data. Finally a graphical user interface (GUI) is provided for easy access to the proposed feature modeling system.

In order to ensure feature rules to be correctly added to the library, a feature rule manager is provided for the designer to insert, delete, and modify feature rules manually. It is also responsible for the resolution of possible conflicts, inconsistencies, and redundancies that may occur between rules due to manual input. Conflicts occur when the same premises are used to match different feature types. It happens due to a failure in realizing that the same premises have already been used to match another feature type. Redundancies mean that the same rules have been defined in the feature library more than once. An inconsistent rule indicates a mismatch between premises and their feature. For example, there is an inconsistency when three contiguous faces have been used to define a feature called **step** which consists of two perpendicular faces.

Conflicts and redundancies can be avoided in the feature library by checking a newly added feature rule against existing ones. To check consistence of a feature rule, two meta-rules are built to filter newly created feature rules. The first meta-rule is the number of logical faces for a feature pattern. For instance, if a key word **step** appears in a feature name, this rule must contain two faces. If the key word **step** is appears to be accompanied by the key word **blind**, this rule must have three faces. If this meta-rule does not fit the rule to be added, the rule manager will ask the user to confirm the correctness of this new rule before storing it. Through

this meta-rule, each time a new feature rule is to be added, the rule manager will check its feature surface to ensure that all these feature faces are in contact with each other. The product modeler is further detailed in Appendix C.

5. PRODUCIBILITY EVALUATION

A typical producibility definition follows a manufacturing difficulty or a manufacturing cost theme. The difficulty theme is followed in this study. For component designs, there are two types of evaluation approaches: qualitative and quantitative. A qualitative approach performs a feasibility check on a design. It usually applies manufacturing rules to a product design and evaluates the producibility on a pass/fail basis. The amount of manufacturing resources required and the level of difficulty are not measured. This evaluation provides the designer a quick overview of a product design and helps eliminate undesired designs at the early modeling stage. The technical feasibility evaluation is usually conducted with in-house manufacturing resources only, rather than on an absolute basis. Manufacturing rules are abundant in various manufacturability handbooks. A quantitative approach, on the other hand, conducts an analytic study of the design and generates a numerical value to report its producibility. Usually it is applied only after the design passes the feasibility check. The numerical value may be used by the designer to improve a design, evaluate alternative designs, or identify the best manufacturing process.

The producibility evaluation is carried out in two phases in this implementation. The first phase is a qualitative evaluation of the design. It checks the technical feasibility at part level and feature level. At part level, it verifies the availability of the material stock and check machine and tool's capability of handling the stock size and weight. At the feature level, it verifies the availability of feasible tools and machines for each form feature, the interference between features, and the compatibility of feasible machines and tools in combination. The evaluation of cutting tools is focused on their cutting capability. The evaluation of machines is focused on the work envelop to accommodate the workpiece. In addition to machine/tool compatibility, it checks the combined effect of machine and tool on processing the workpiece. For example, a tool on a machine may reduce the work envelop. Tool clearance and the *thin wall* effect between two depression features are both considered.

The second phase is focused on an evaluation of the product design with a set of producibility measurements developed in this study. The producibility index is calculated based on the compound effect of the following four measurements: 1) the ease of machining of the stock material, 2) the ease of handling from the stock size point of view, 3) the ease of handling from the stock weight point of view, and 4) form feature complexity. The ease of machining is calculated with the machinability index obtained from the TMEH machinability databook. The size aspect of handling ease penalizes only a small-sized design, leaving the weight factor to capture the oversize issue because presumably the design has passed the feasibility check. Similarly, the weight aspect of handling ease penalizes only a design with heavy weight, leaving the size factor to capture the light weight impact, because again the design is assumed to have passed the weight limit check.

At the feature level, the system evaluates each form feature in terms of its feature type, feature orientation, surface finish, and tolerance. The penalty assessed for each feature is accumulated into an integrated feature penalty index, converted into a feature producibility, and then incorporated into the overall producibility index. All producibility indices are ranged between 0 and 1 for easy interpretation. An index value of 1 implies no manufacturing difficulty. An index of 0, on the other hand, means that the manufacture of this design is extremely difficult. The producibility evaluator is described in detail in Appendix D.

6. PROCESS PLANNING

Process planning is an engineering activity which translates a product design into a manufacturing process. A process plan usually details each sequenced operation required in the process and its associated machine and tool on a route sheet. Some may contain detailed machining parameters, fixtures, and setups. Often it may be also desirable to specify the shape and dimensions of the material volume to be removed at each step. The process planner developed in this project generates a process plan for a machined part design in three stages. The first stage converts design form features from a product model into a set of sequenced material removal volumes (MRVs). The conversion of design features into manufacturing features is carried out in four steps. The first step converts a design feature into a temporary MRV. The second step evaluates the status of each open faces on each MRV. The third step refines each

temporary MRV into one or multiple (final) MRVs such that each MRV has either no island or island(s) with the same height as the refined MRV's. The fourth step sequences and groups MRVs into a feasible machining sequence.

The second stage searches for a feasible machine and cutting tool(s) for each of the sequenced MRV. To improve the efficiency of the machine and tool selection process, a feature/machine/tool table is created for each manufacturing feature (MRV) to record its possibly feasible machine and tool combination(s). Corresponding to each feasible machine and tool combination, a knowledge module for process planning has been developed for selecting tool and machine instances within its class for the given form feature. Each knowledge module defines tool and machine specifications for this form feature and its pre-processing requirements. Therefore the search of machine and tool for a form feature may cause a sequence of machining operations being identified as its pre-processing requirements.

The third stage searches and evaluates alternative plans for process planning optimization. A new process plan can be created by using a different MRV sequence, selecting a different machine, or using a different tool. To improve the search efficiency and avoid redundant plans, a tracking mechanism was implemented which keeps a record of the search and points to the next. The quality of a process plan is evaluated according to the objective specified for the planning session. The final process plan is organized into a sequence of sub-processes, each of which is accompanied by a workpiece setup requirement and an NC program. A subprocess may have one or multiple operation clusters. Each cluster is a collection of the machining operations required to create a manufacturing feature (i.e., an MRV). An operation is the lowest level machining activity and is qualified by the requirement of one cutting tool and accompanied with an elemental MRV. A detailed description of the process planning is provided in Appendix E.

7. NC PROGRAM MODELING

NC program modeling prepares a set of G codes required to control a selected machine for removing one or multiple MRVs from the workpiece in one workpiece setup. Typically NC programming is done manually or more recently with help of a user-interactive CAM system. In either case, the user needs to describe the geometry and dimensions of the part design, specify machining parameters, and determine tool path. The tool path file is then translated into G-codes

according to the format specified in the selected machine controller. The NC programmer implemented in this project seeks automatic translation from an MRV to NC codes for a selected machine controller, without manual input of the above data. The system first retrieves a process plan and identifies each subprocess requiring an NC machine. The cutting tool and its associated elemental MRV are identified to match a canned cycle. If a canned cycle is found, the system needs to only prepare the required machining parameters for the canned cycle. In case that the MRV matches several canned cycles, a decision rule is applied to select the most appropriate canned cycle. It often occurs in a hole-drilling operation.

If there is no match, which may be typically the case, a generic canned cycle program is then used to calculate machining parameters and tool path, and convert the cutter location file into NC codes using primitive G commands specified by each machine controller. After all clusters in a subprocess are processed, the system compiles an NC program for each subprocess and attached to the process plan. A set of machining parameters are also extracted and saved along with the NC program to support other engineering and evaluation activities such as cost estimation. The development of NC programming is further detailed in Appendix F.

8. IMPLEMENTATION

This concurrent engineering system has been coded in SUN C++ version 3.0.1 and run on SUN/OS version 4.1.3 on SUN/SPARC 10/20 stations. It requires the ACIS geometric modeler (currently version 1.5) and the VERSANT object-oriented database management system (currently, version 3.0.8). The ACIS supports product modeling and the VERSANT stores manufacturing data and knowledge. The graphic user interface (GUI) for product modeling was developed with the X Window 11R5 library. Other GUIs including the ones for process planning, producibility evaluation, NC program modeling, manufacturing knowledge manager, and the manufacturing resources manager, were all developed with the XVIEW 2.0. The implementation of the product modeler and the producibility evaluation are presented in their respective appendix. This section describe, from a user point of view, the implementation of the process planner, the NC programmer, the manufacturing resources manager, and the manufacturing knowledge manager.

8.1 Process Planner

The process planner converts design features into material removal volumes (MRVs), arranges the MRVs in a sequence feasible for machining operations, selects machine and tool instances to process these MRVs, and identifies the optimal operation sequence with respect to a given criterion. A graphic user interface (GUI) developed in XVIEW for the process planner is shown in Figure 2, which organizes various planning functions for the planning system. Each function button has two operation modes. An inactive function button has a dimmer color and cannot be clicked by the user. Only active buttons are available for activation.

The *SETUP* button allows the user to specify an objective for this planning session. The default planning objective is minimum number of machine setups. On the left margin of the window, a list box is provided to list all product designs in the database. In this example, the product #7 is selected; its header information is displayed on the window. Once a product design is selected, the buttons *NEW* and *RETRIEVE* become active. The command *NEW* generates a new process plan; and the command *RETRIEVE* retrieves all existing process plans from the database. If multiple plans exist for the selected product design, an item selector with an increment and a decrement button appears on the right hand corner of the window. The ID of the current process plan on display and the total number of process plans in the database are also displayed. The user locates the desired process plan by clicking the increment or decrement button.

Once a process plan is selected, the buttons *APPROVE, SAVE AS, PRINT*, and *DELETE* buttons become active. To approve a process plan, an authorization ID and the password are requested and verified. After verification, the approver's name and approval date are stored along with the process plan. An approved plan can not be re-approved. The *DELETE* button is provided to delete a process plan from the database. An approved plan can be deleted by its approver or a super user. The *PRINT* button is provided to create a hard copy of a process plan. The current plan can also be saved into a text file for further reference by clicking the button *SAVE AS*. The MRVS button allows the user to bring up a sub-window, shown in Figure 3, which displays the details of manufacturing features (MRVs) and related data. These data were generated during the first stage of process planning and are not persistent objects. Therefore they

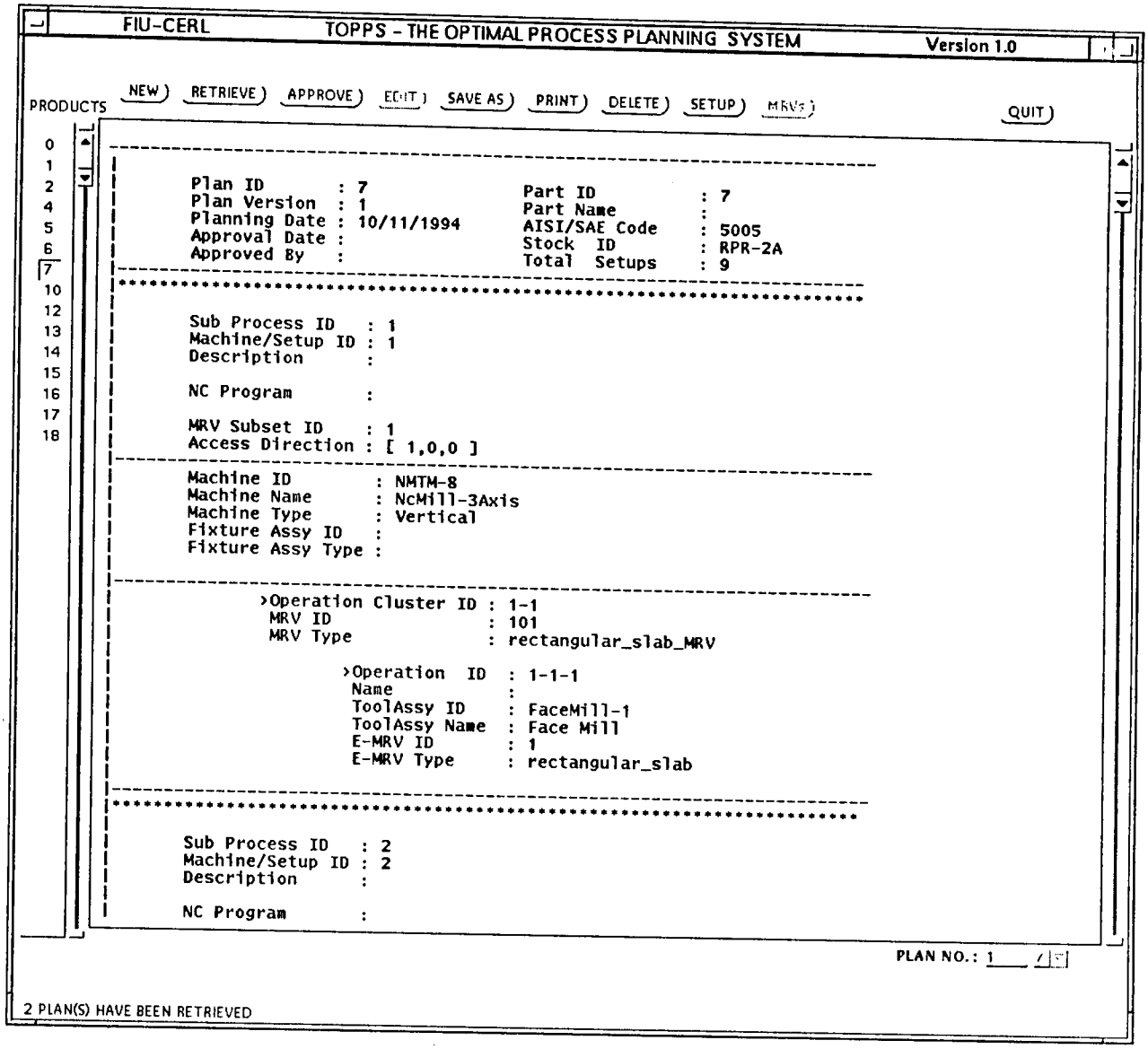


Figure 2: GUI for Process Planning

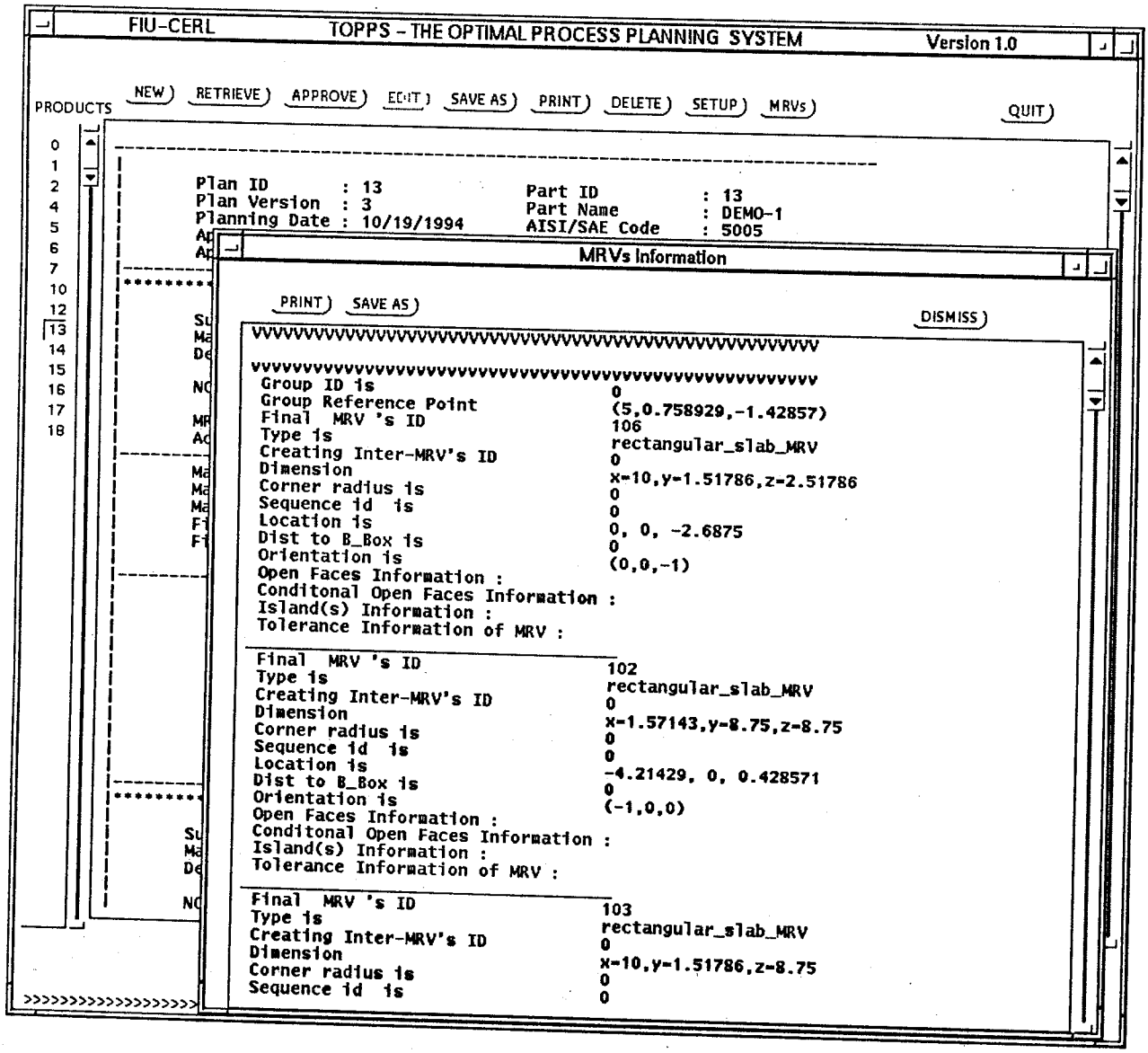


Figure 3: MRV Subwindow

are not stored in the database. The *QUIT* button is provided to quit the system.

8.2 The NC Program Modeler

The NC program modeler transforms a process plan into CNC codes according to the machines and tools specified in the plan. This system interface provides five functional commands for NC programming. The user can retrieve, create, edit, save, and print an NC program. The system requests the user specify a product design to retrieve its process plans. All buttons are activated after a product design is selected. If there are multiple approved process plans for a product design, a plan selector with an *increment* and a *decrement* button will appear at the right hand corner of the main window. By clicking the buttons, the current selection will be displayed on the screen. After a plan is selected, NC codes are generated according to the operations specified in the process plan. After NC codes are generated, the buttons *EDIT, SAVE AS, PRINT* and *DELETE* become active. By clicking the *EDIT* button, the editing mode is activated for manual NC codes modifications. After editing, the system invokes an NC code syntax verifier to check syntax errors in the NC program. The SAVE AS button is activated to save an NC program and a link is built between the program and the process plan. The button *QUIT* is used to quit an NC program generation session.

8.3 The Manufacturing Resources Manager

The manufacturing resources manager provides a friendly interface between the user and the VERSANT database management system to control manufacturing data stored in the database which currently includes machines, tools, fixtures, and material stocks. This comprehensive and integrated database manager keeps these data consistent and up to date; and allows the user to add, delete, browse, update, and print manufacturing data stored in the database. Figure 4 shows the menu hierarchy of this manager. The button *ADD* is provided for adding new data to the database. With *ADD* being activated, a sequence of menus will be presented until it reaches the target menu. A scroll bar is provided as necessary to select a desired item on the window. Figure 5 shows a sub-window for updating a tool instance under the class hierarchy of *tool/millTA/Mdrill*. The user can use the back space bar to erase an attribute value and re-type

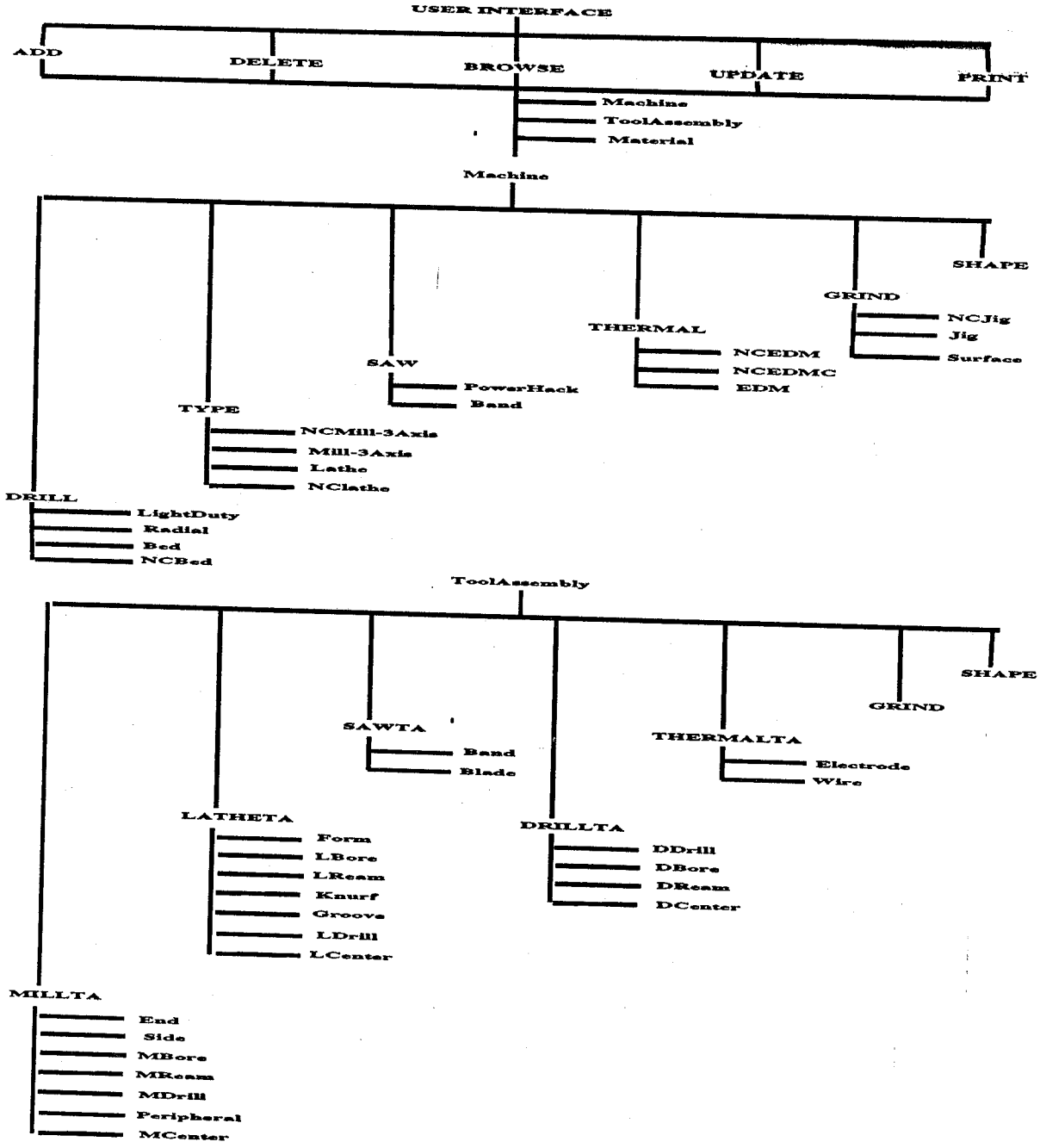


Figure 4: Database Manager's Menu Hierarchy

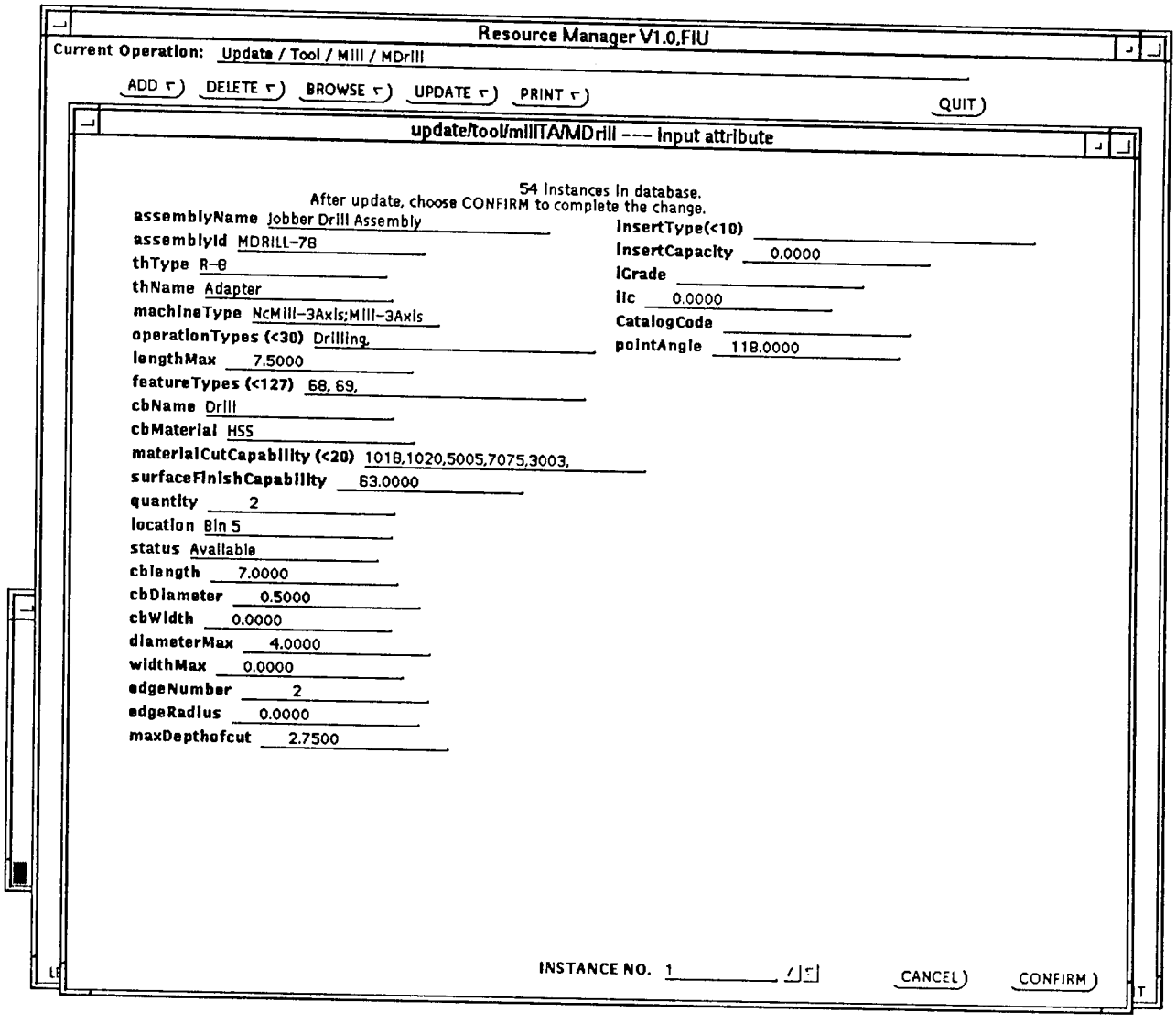


Figure 5: Tool Instance Update Subwindow

the new one. The CANCEL and CONFIRM buttons are provided to cancel or confirm an action. Similar to *ADD* and *UPDATE* buttons, other commands including *DELETE*, *BROWSE*, and *PRINT* were similarly designed.

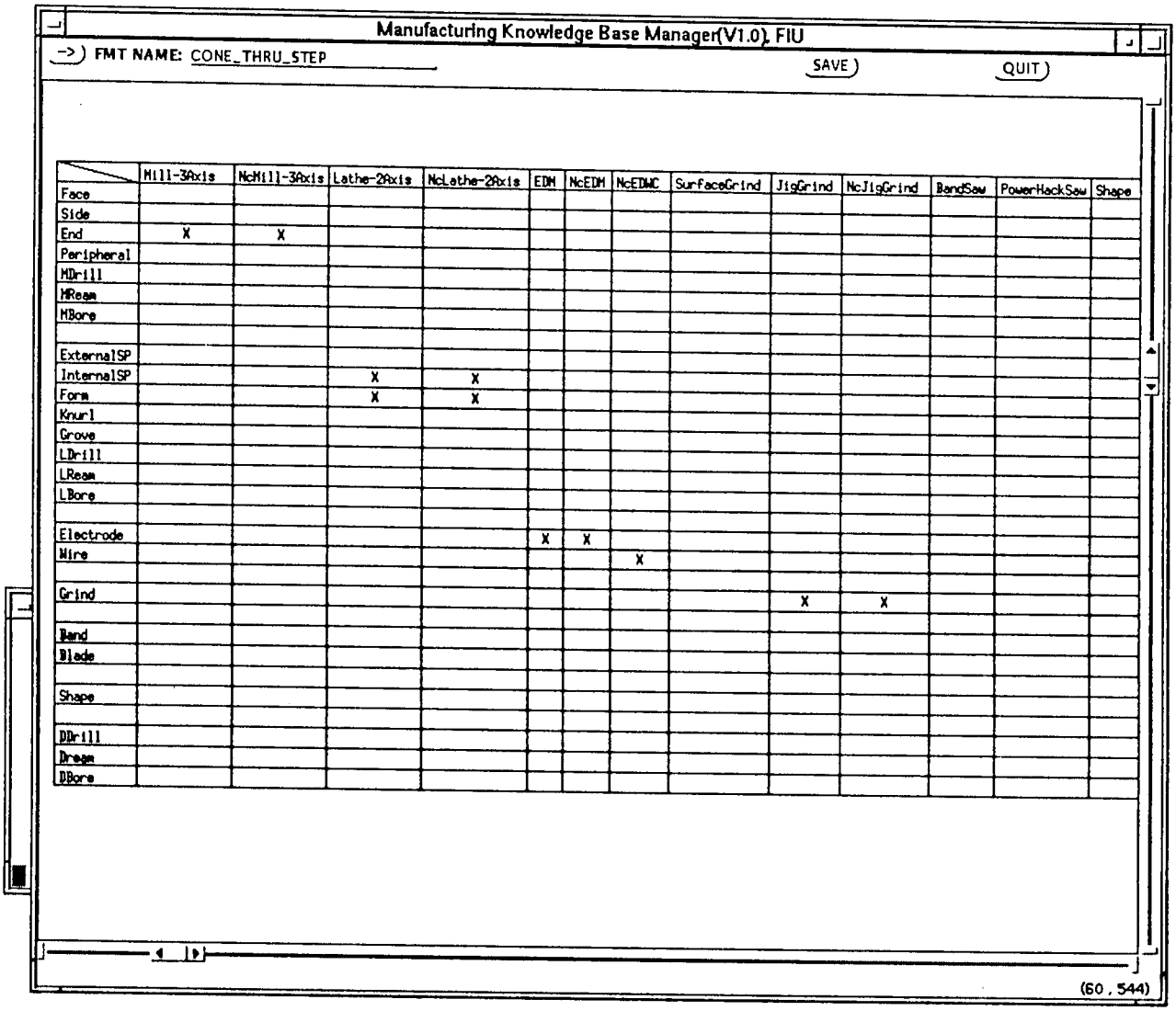
8.4 The Manufacturing Knowledge Base Manager

The manufacturing knowledge currently captured in the system are:

- rules for feature recognition;
- rules for operation selection;
- rules for machine and tool selection; and
- rules for machining parameter selection.

The knowledge base manager allows the user to view, add, delete, modify, and print the knowledge. The main window layout of the manager is presented in Figure 6. Shown on this window is the F/M/T table for a manufacturing feature called *CONE-THRU-STEP* MRV. To select another F/M/T table, the button "->" is provided on the upper left corner of the window. By clicking it, the system will pop up a list box with a scroll bar to display all feature types currently supported. In the list box, the currently selected feature name is shaded. To view the F/M/T table for another feature, a scroll bar is provided to move the feature list. When the right one is located, the left button on the mouse can be clicked to select and the shade will show on the selected feature to confirm. After confirmed, the display of the F/M/T table is updated for the newly selected feature.

Only authorized users are allowed to modify F/M/T tables. To add a machine and tool combination to the current table, the cursor is used to locate the appropriate cell in the table. With a click on the mouse's left button, a check mark, "X", appears in the cell to signify the selection. The same procedure is applied to delete an entry from an F/M/T table. To make a modification permanent, the *SAVE* button is provided at the upper right corner of the window to save all modifications into the database. The *QUIT* button is provided to abort an editing session.



	Mill-3Axis	NcMill-3Axis	Lathe-2Axis	NcLathe-2Axis	EDM	NcEDM	NcEDWC	SurfaceGrind	JigGrind	NcJigGrind	BandSaw	PowerHackSaw	Shape
Face													
Side													
End	X	X											
Peripheral													
MDrill													
MReam													
MBore													
ExternalSP													
InternalSP			X	X									
Form			X	X									
Knurl													
Grove													
LDrill													
LReam													
LBore													
Electrode					X	X							
Wire							X						
Grind									X	X			
Band													
Blade													
Shape													
DDrill													
Dream													
DBore													

Figure 6: Manufacturing Knowledge Manager

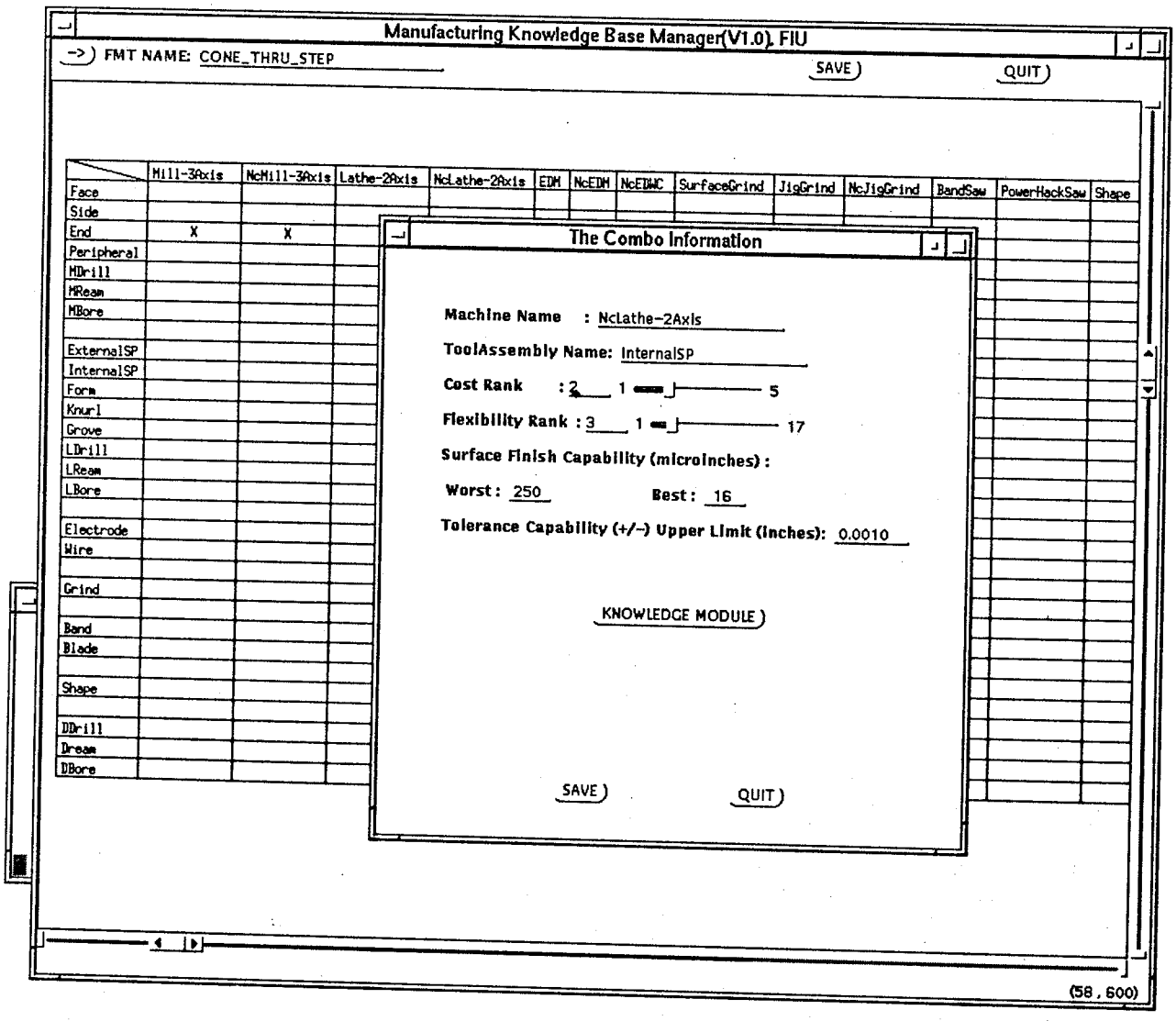


Figure 7: Machine and Tool Class Combo information

Each check mark in an F/M/T table corresponds to a manufacturing knowledge module in the database. The structure of the knowledge modules is detailed in Appendix G. Each module corresponds to a machine and tool class combination (COMBO). Figure 7 shows the attributes defined for the current COMBO, which includes the machine class name, tool assembly class name, relative cost rank, flexibility rank, surface finish range (the worst and the best), and tolerance capability. These attributes are also defined in Appendix G. Only authorized users can change these attributes. The *SAVE* button allows changes to be saved into the database permanently. The *Knowledge Module* button provides an assess to the content of each COMBO's knowledge module, which is detailed in Appendix G.

9. INDUSTRIAL COLLABORATION

The collaboration with the industry has been an emphasis during the entire project period. To solicit industrial interest, this research project was first publicized in a PDES/STEP meeting and an expression of interest was received from several companies including Lockheed, McDonnell Air, Concurrent Engineering, Pratt and Whitney (P&W), and Ford Motors. After careful evaluation, the last two were selected as pilot sites. They were selected for geographic proximity and product diversity. In the past two years, several other companies have also expressed an interest in this project. Some companies such as Texas Instruments and Standard Technology have sent representatives to visit FIU while others such as Cognition and NCMS invited the principal investigator to make a technical presentation at their facilities. Several academic institutions including the Penn State, Texas Tech, and George Washington also expressed an interest in building their research and development on our research project.

In this project period, a close working relationship was established between FIU and P&W at West Palm Beach, Florida. There has been frequent visits between these two parties. P&W made its contribution to the project by providing their industrial viewpoint and technical (manufacturing and design) expertise. Through many presentations and demonstrations, FIU made contribution to P&W by providing the R&D update, analysis of its impact to the company, and assessment of its CAD/CAM systems in use. Even through P&W is highly interested in this project, many difficulties have been identified as obstacles to transferring R&D results from an academic environment to an industrial company such as P&W for routine use. These difficulties

are: 1) P&W would feel much more comfortable with a commercial package from a company which has a 1-800 number for maintenance and other services, 2) P&W does not have the ACIS and the VERSANT packages which are required to run the current system developed in this project, 3) there is no data translator between this system and the CAD and CAM packages currently in use at P&W, and 4) P&W at West Palm Beach, FL., is not actively engaged in production. Despite of the above difficulties, the plan for further system testing and transferring this system to P&W still continue, even though the research project expired last August. In early September, the FIU research team made a one-day visit to P&W to collect manufacturing resource data and finalize the selection of sample parts for a walk-through test and demonstration at Pratt & Whitney in this fall.

The FIU research team has also established a contact with investigators at NIST to test this system at the process planning testbed in NIST. In the meantime, the research team has constant contacts with engineers at Ford Motors and Texas Instruments. One of the research collaboration concepts is on producibility evaluation in sheet metal application in these two companies. In addition, some universities would like to use the product modeler developed in this project to further their research activities in process planning and to use the process planning system to study inspection planning. The FIU research team is therefore actively making the system independent of third party software systems and moving this system to a PC/NT platform to meet the needs of these universities.

Many presentations on this research project and demonstration of this system have been made at technical meetings. Several papers on this subject have been included in conference proceedings. One paper has appeared in a technical journal. Another paper has been accepted by a technical journal. Three additional journal papers are currently in review. Four more journal papers are in preparation. These papers are listed in Appendix H.

10. CONCLUSION

The objective of this research project was to develop a feature-based system which enables a product to be designed concurrent by a continuous check on its manufacturability. The specified tasks to be addressed were identified as: 1) to classify form features on mechanical part designs and to capture part designs in an object-oriented product information model, 2) to

develop a feature-based producibility assessment model, 3) to develop (or extend) a feature-based product modeling system, and 4) to integrate the assessment model with the product information model. The goal of this research project is a methodology for developing a feature-based and object-oriented engineering system for concurrent product and process design of machined components.

All the tasks identified above have been accomplished in this research project. The goal of establishing a methodology for developing a feature-based engineering system for concurrent product and process design has been reached. Several engineering application modules were developed to validate the current engineering environment. This research project has made a significant contribution to the CAD/CAM research community in the areas of manufacturing resources modeling, manufacturing knowledge modeling, form feature classification, product modeling, process planning, producibility evaluation, and NC programming.

Many additional engineering applications can be built and added to the concurrent engineering environment. Material flow simulation and production planning modules should be included to widen the capability of this system. The producibility evaluator can be readily extended to estimate product quality, manufacturing cost, and production schedule compatibility. Fixturing evaluation in setup planning is another area for extension. In the current implementation, the system only selects from machine vises to verify setup feasibility. No modular fixturing or custom-made fixtures are considered. In product modeling, meta-rules may need modifications to handle more complicated form features. The face merging issue also requires further study. The inclusion of non-CSG operations such as sweeping in the modeler is highly desirable. However, how to handle implicit feature volumes is a challenge. In process planning, handling 3D form features and selecting machine tools of higher degrees of controllability are two research issues. The detection of feature interference can also be further improved. The extension of this system's evaluation and planning capability to include other manufacturing processes such as casting, welding, forging, and extrusion is certainly a new research challenge and the extended research should be very fruitful. With this extension accomplished, a product design could be evaluated for multiple manufacturing processes. A process plan which combines two manufacturing processes would also become possible. For example, the process planner would then be able to select a casting process to create the initial

part shape and select a machining process to meet its tolerance and surface finish requirements. The extension of the product modeling system to other types of product designs such as assembled products and sheet metal designs would greatly increase not only the modeling capability but also the scope of downstream engineering applications such as process planning and producibility evaluation of assembly and sheet metal designs.

Appendix A: Feature Definition and Classification

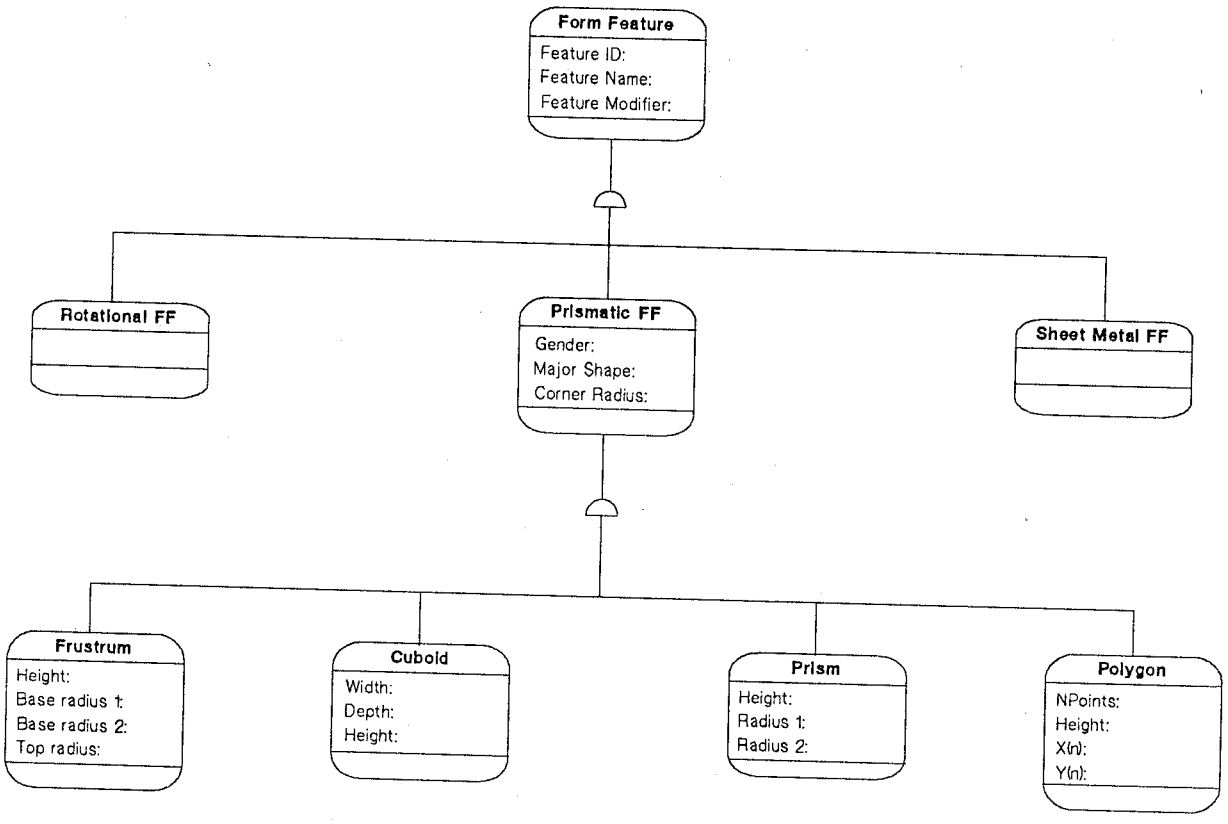


Figure A1: Form Feature Classification

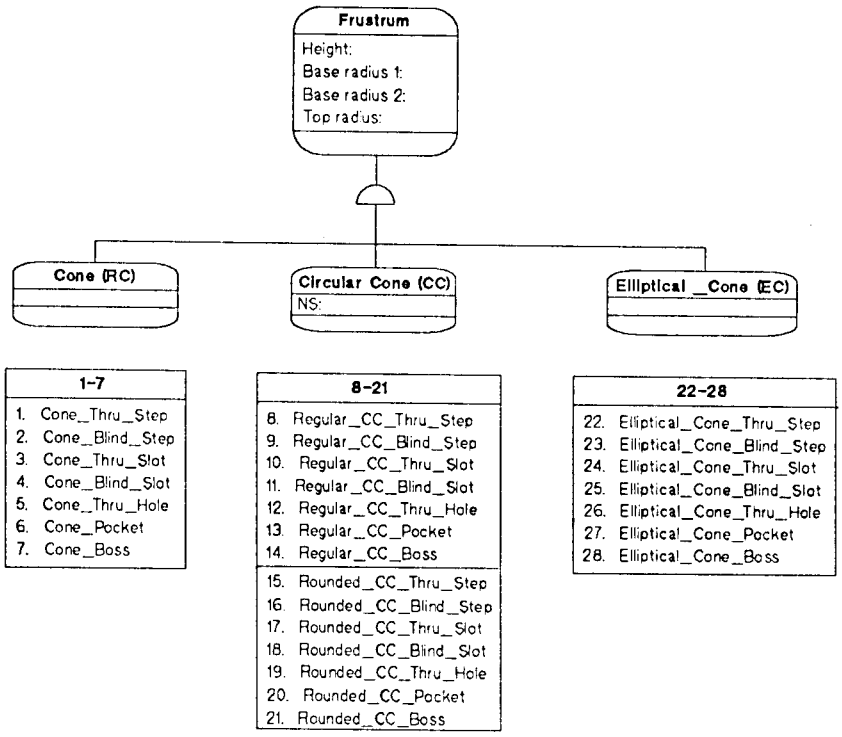


Figure A2: Form Feature Classification
(continued)

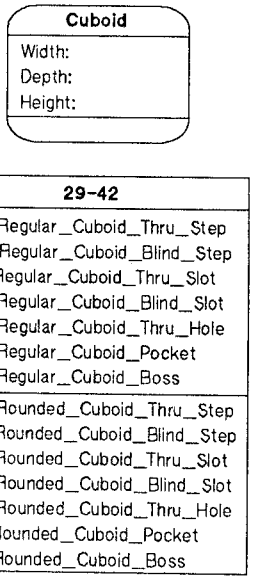


Figure A3: Form Feature Classification (continued)

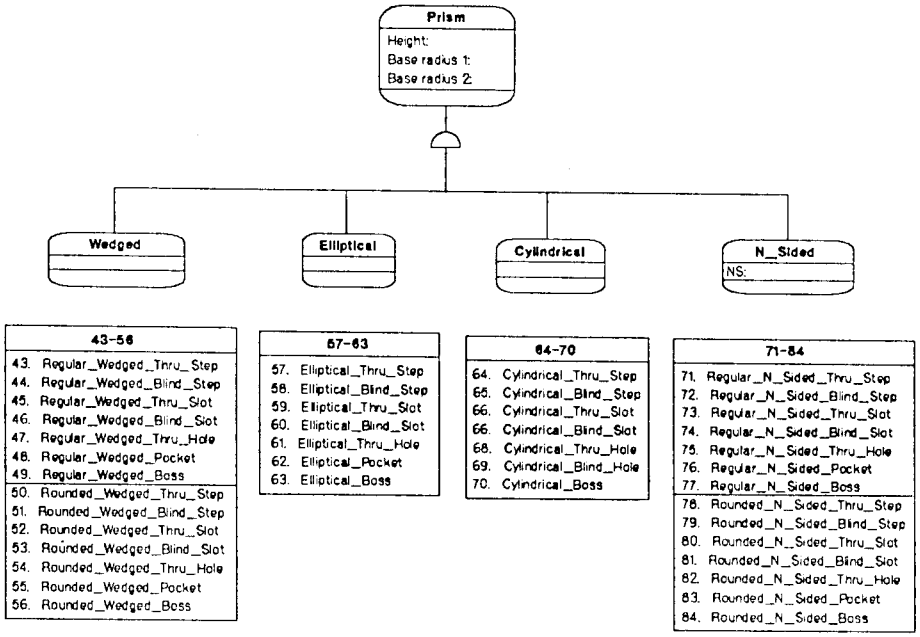


Figure A4: Form Feature Classification
(continued)

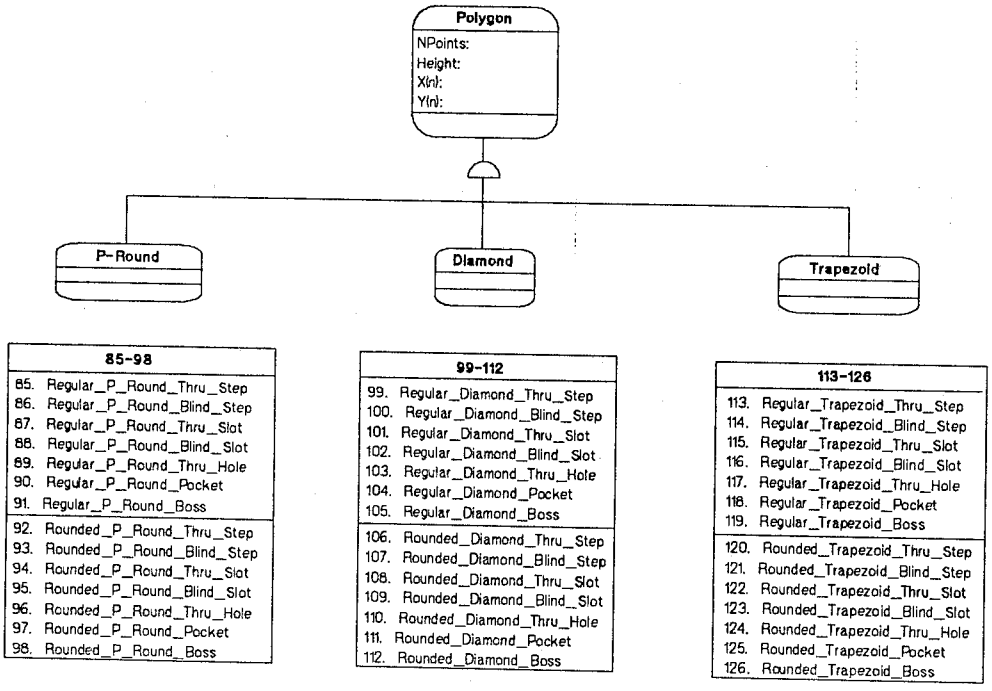


Figure A5: Form Feature Classification
(continued)

(Primitive – Frustrum)

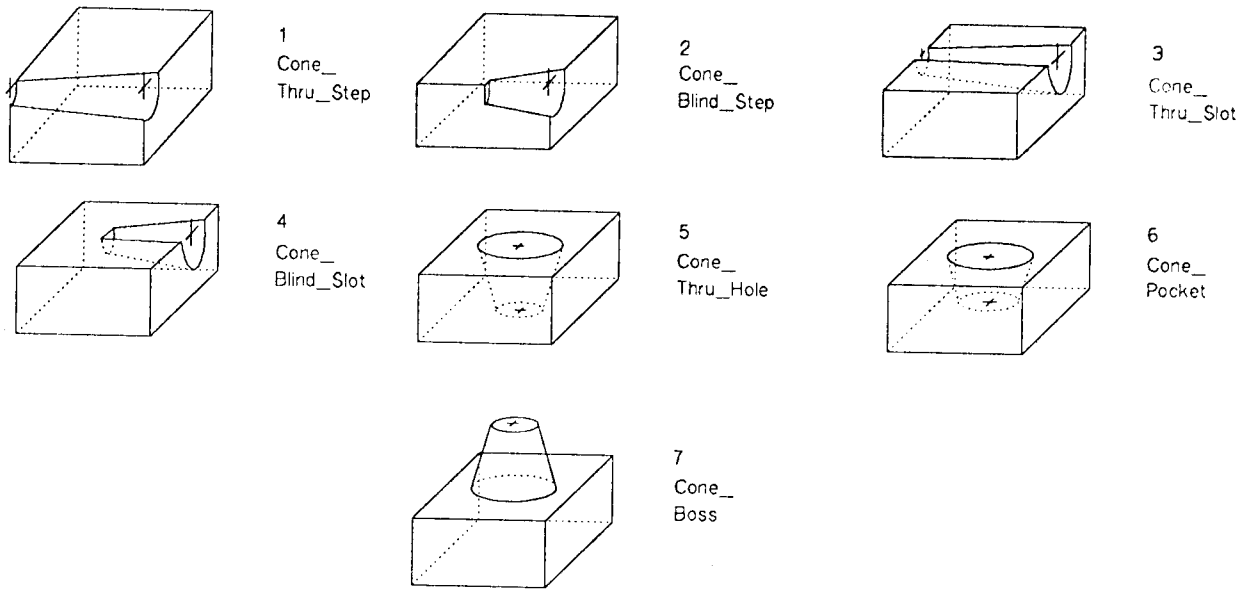
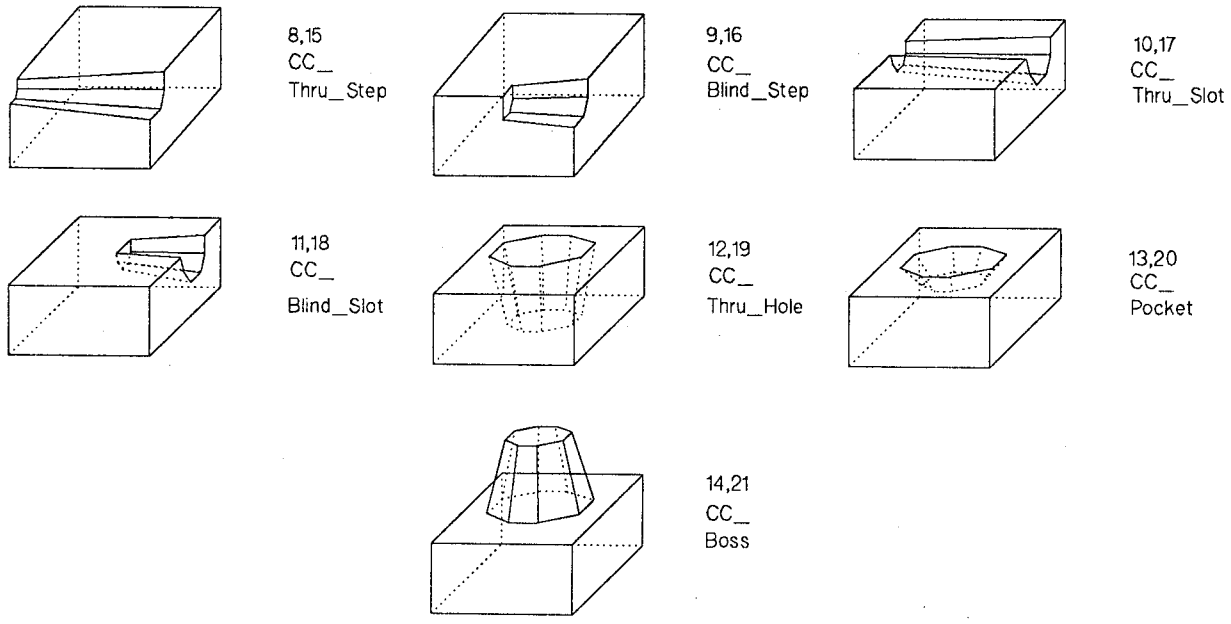


Figure A6: Cone (RC) Class Form Features
(1–7)

(Primitive - Frustrum)



(Regular case shown)
n = 8

Figure A7: Circular (CC) Class Form Features
(Regular,Rounded)
(8-21)

(Primitive – Frustrum)

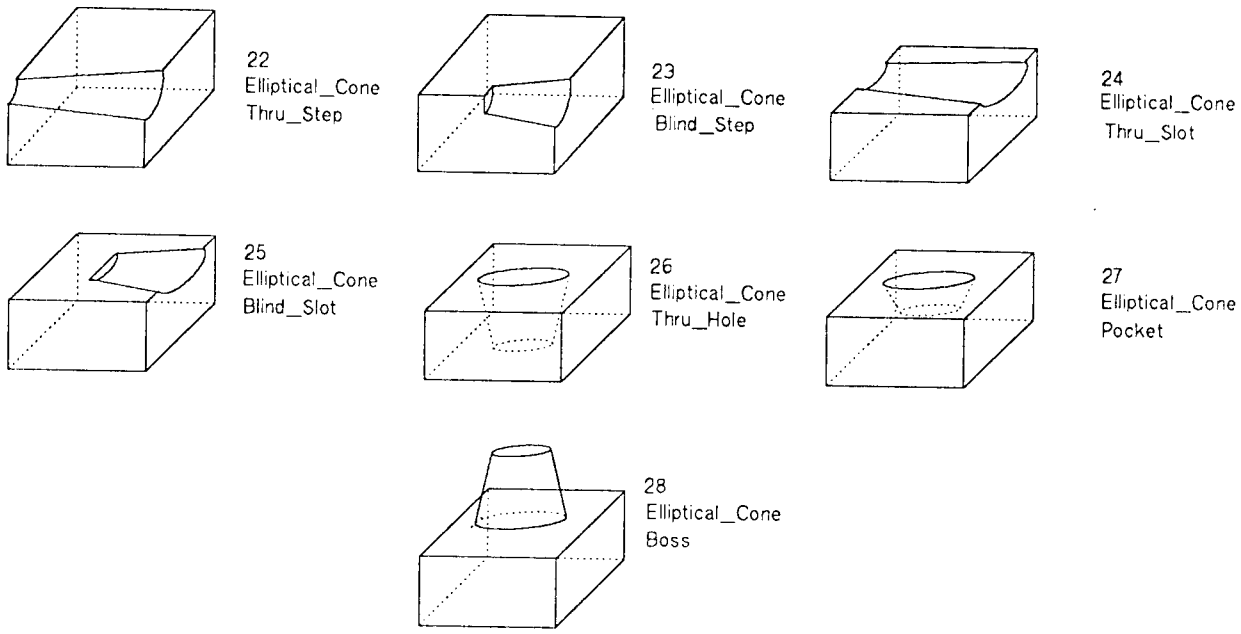
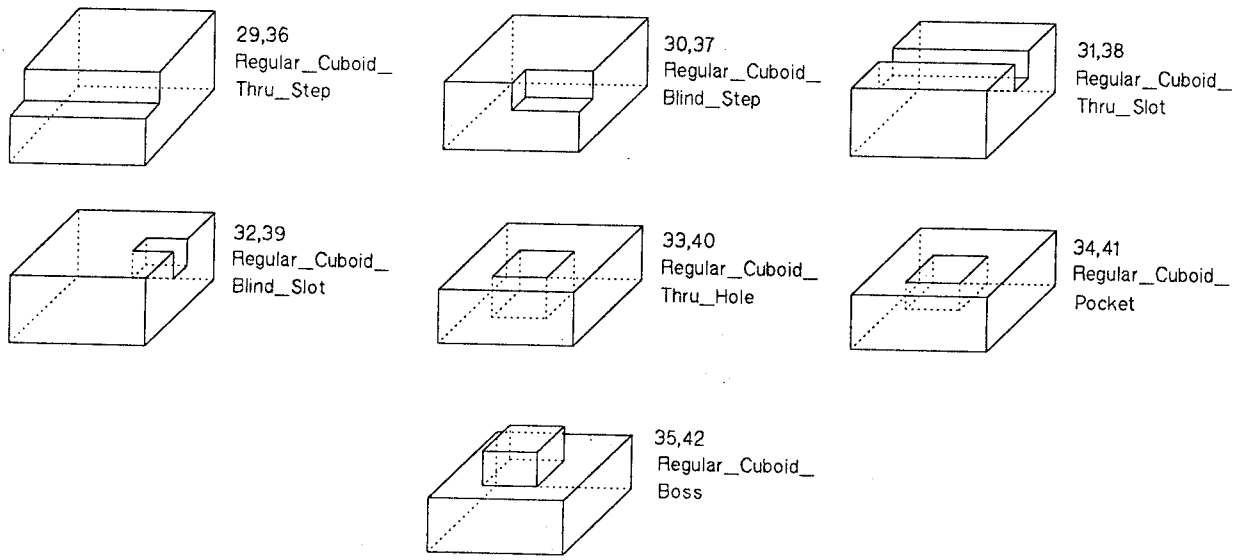


Figure A8: Elliptical Cone (EC) Class Form Features (22-28)

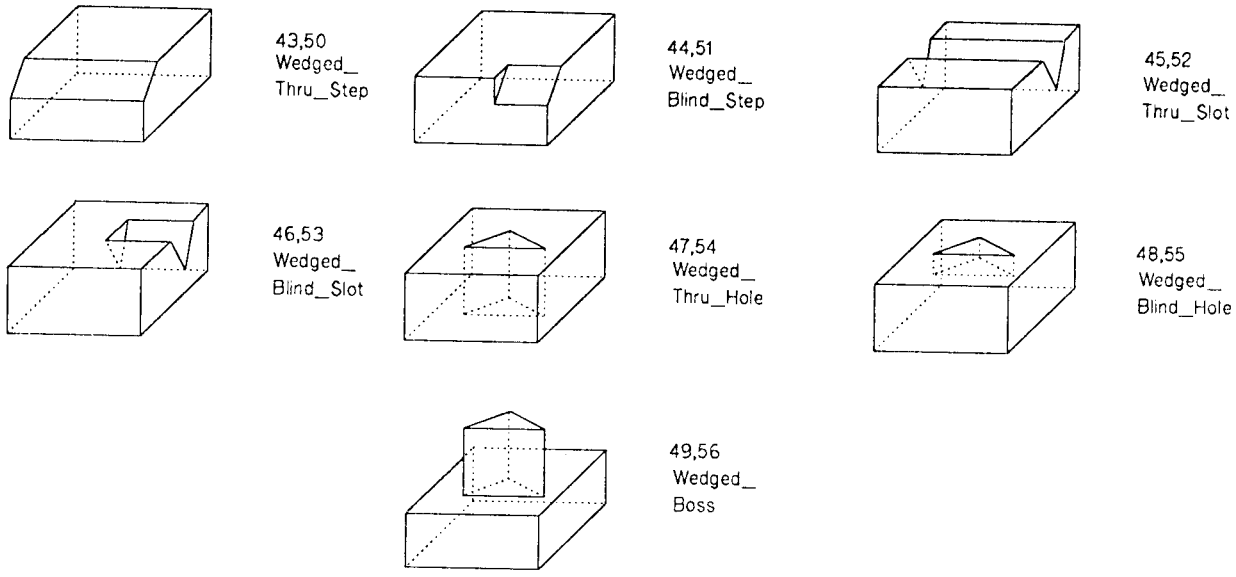
(Primitive – Cuboid)



(Regular case shown)

Figure A9: Cuboid Class Form Features
(Regular,Rounded)
(29–42)

(Primitive - Prism)



(Regular case shown)

Figure A10: Wedged Class Form Features
(Regular,Rounded)
(43-56)

(Primitive – Prism)

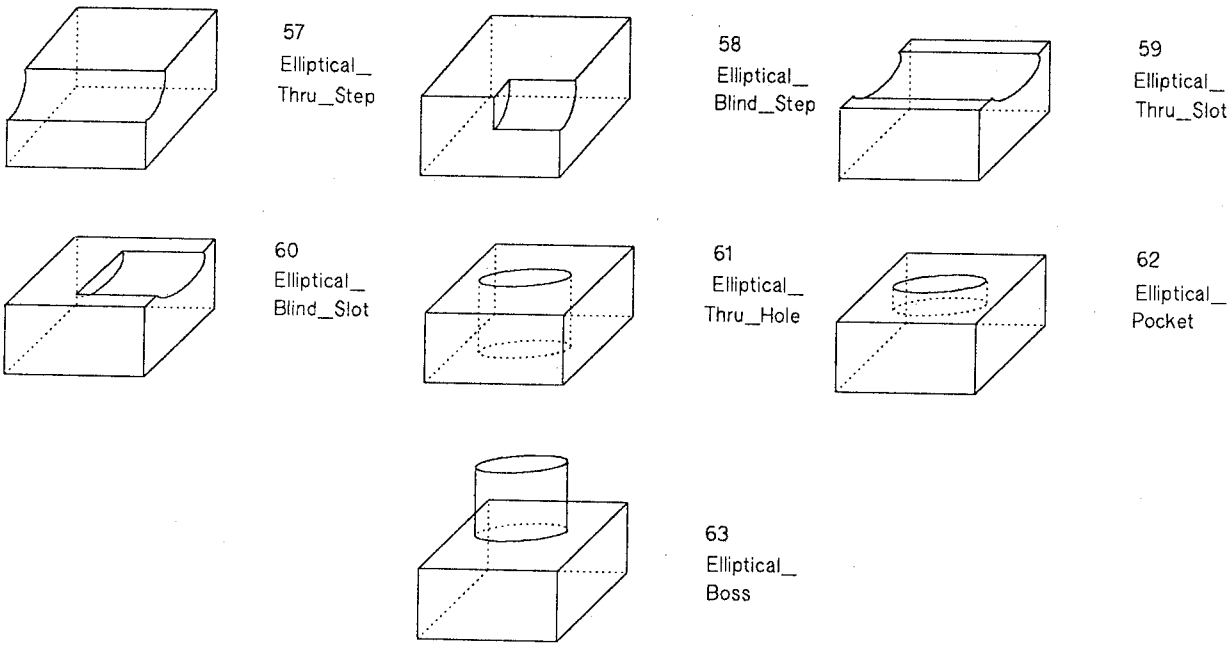


Figure A11: Elliptical Class Form Features (57-63)

(Primitive - Prism)

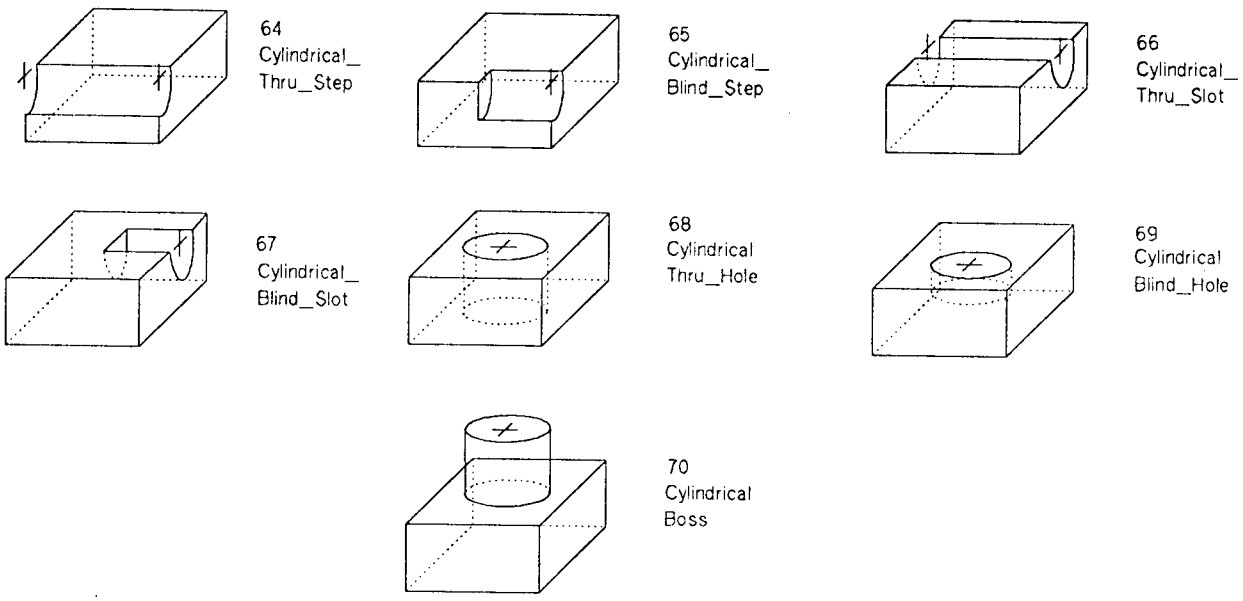
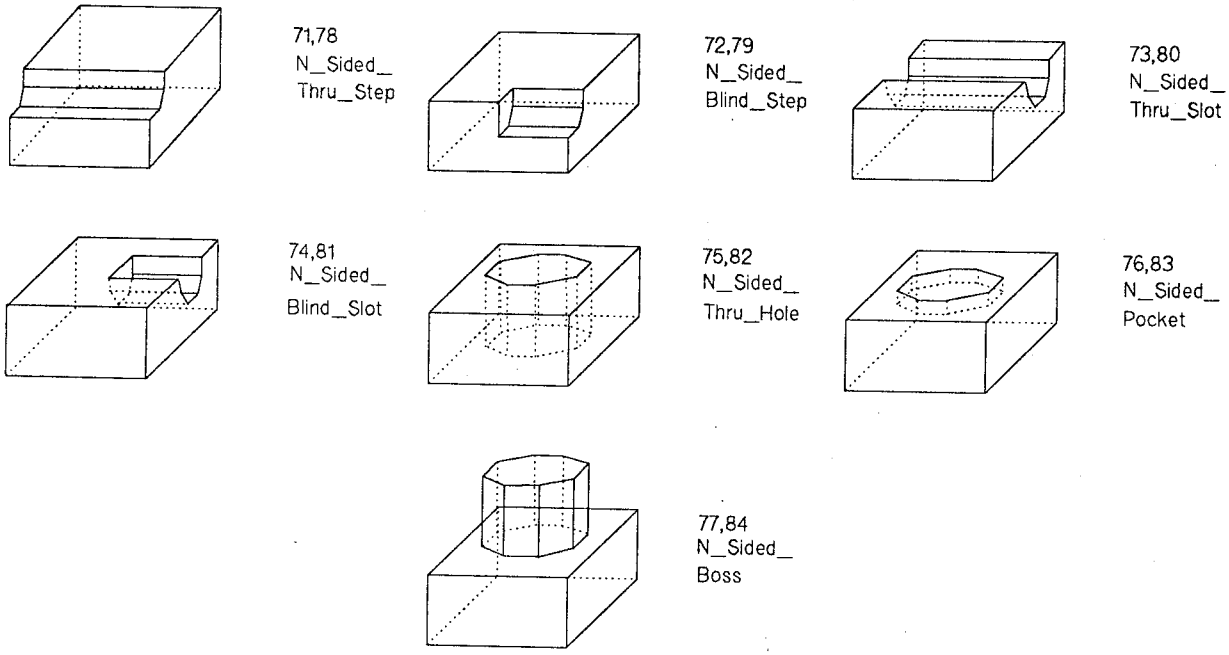


Figure A12: Cylindrical Class Form Features
(64-70)

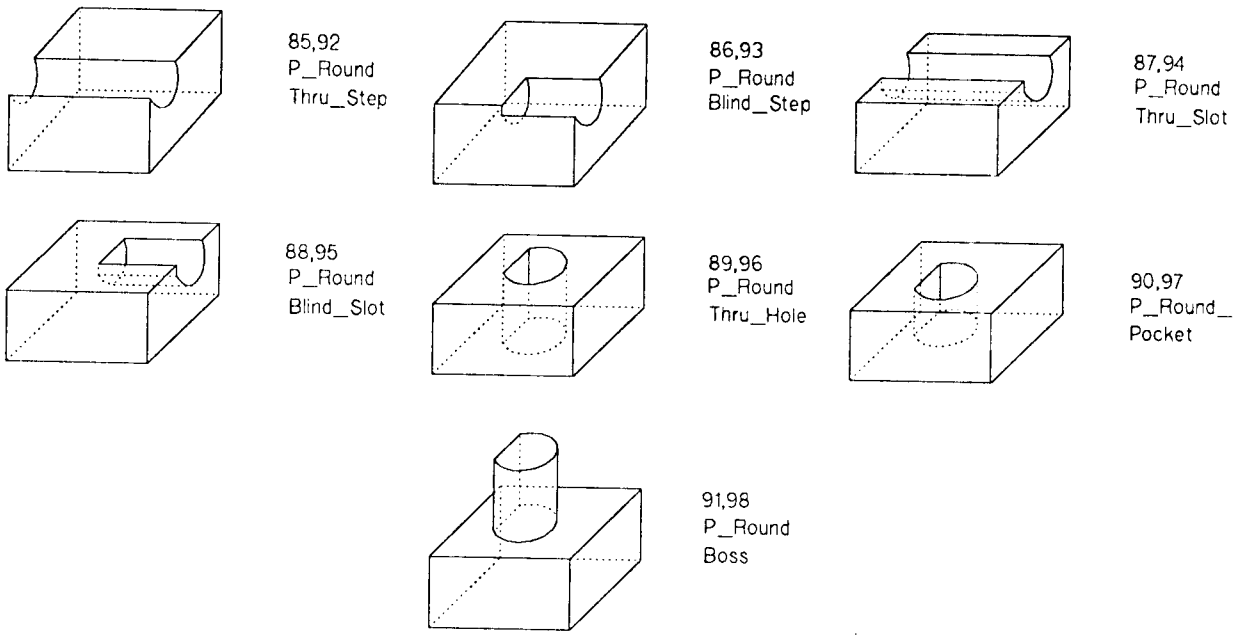
(Primitive - Prism)



(Regular case shown)
n = 8

Figure A13: N_Sided Class Form Features
(Regular,Rounded)
(71-84)

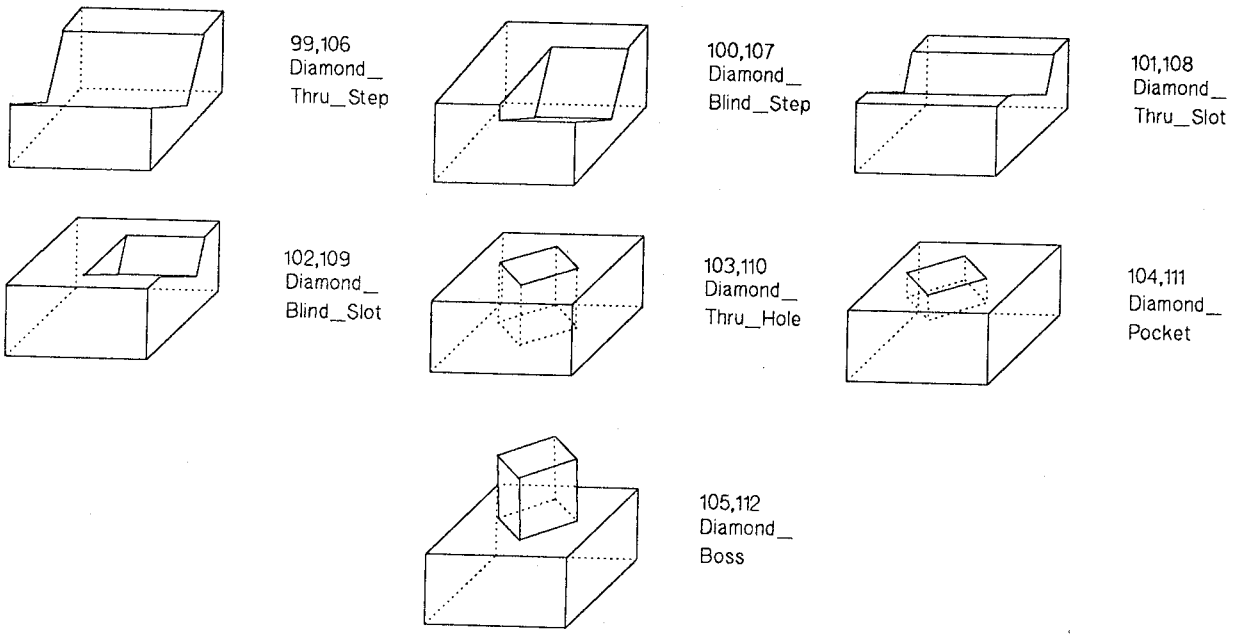
(Primitive - Polygon)



(Regular case shown)

Figure A14: P-Round Class Form Features
(Regular,Rounded)
(85-98)

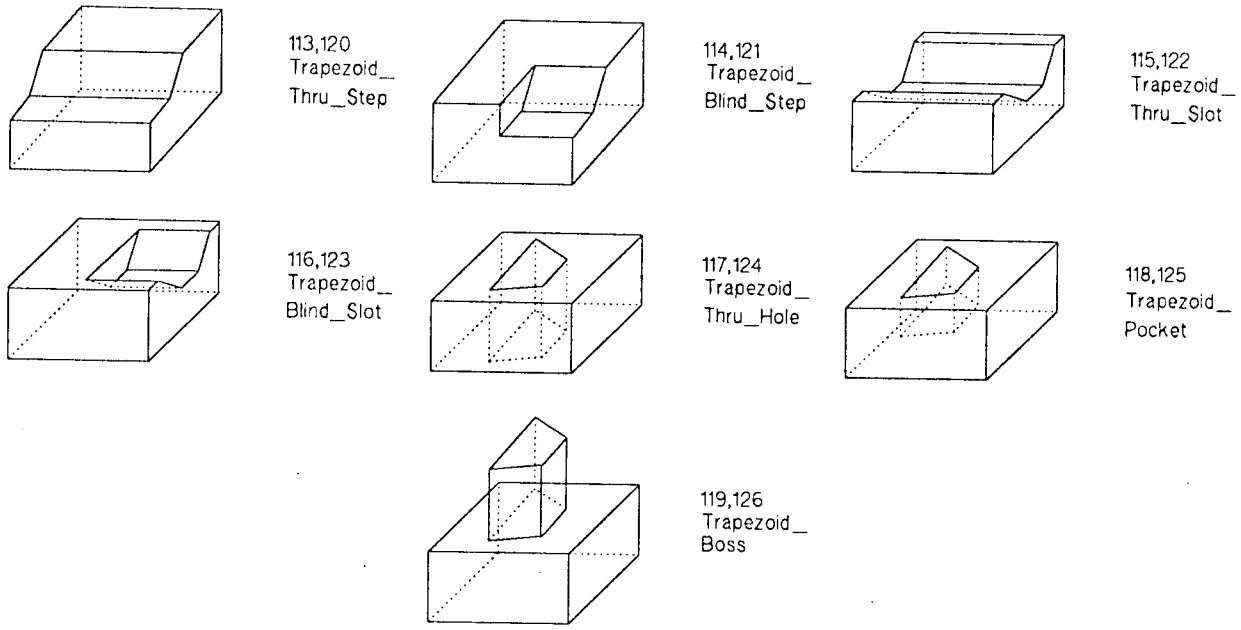
(Primitive – Polygon)



(Regular case shown)
npoints = 4

Figure A15: Diamond Class Form Features
(Regular,Rounded)
(99–112)

(Primitive - Polygon)



(Regular case shown)
npoints = 4

Figure A16: Trapezoid Class Form Features
(Regular,Rounded)
(113-126)

Appendix B: Manufacturing Resources Classification

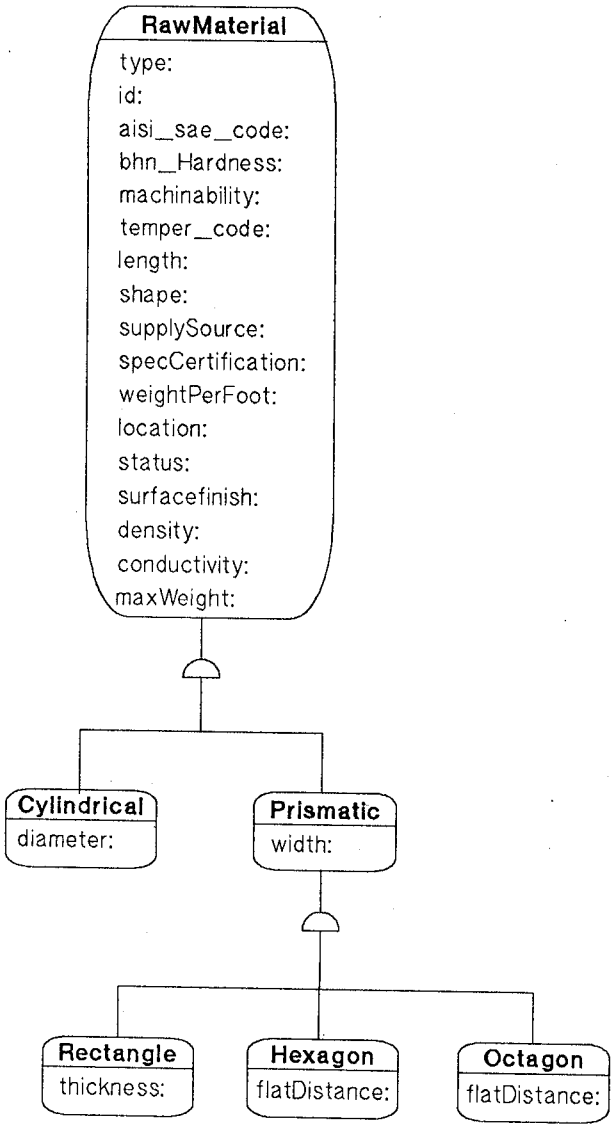


Figure B1: Raw Material Class Structure

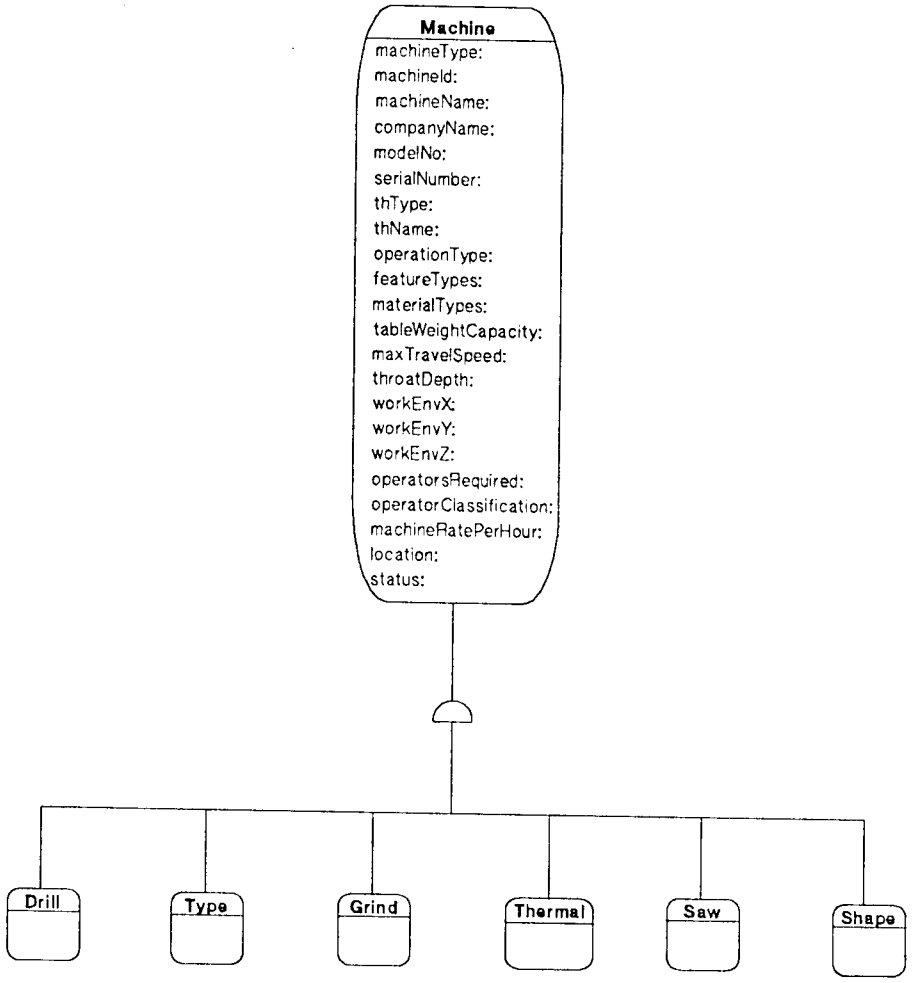


Figure B2: Machine Class Structure

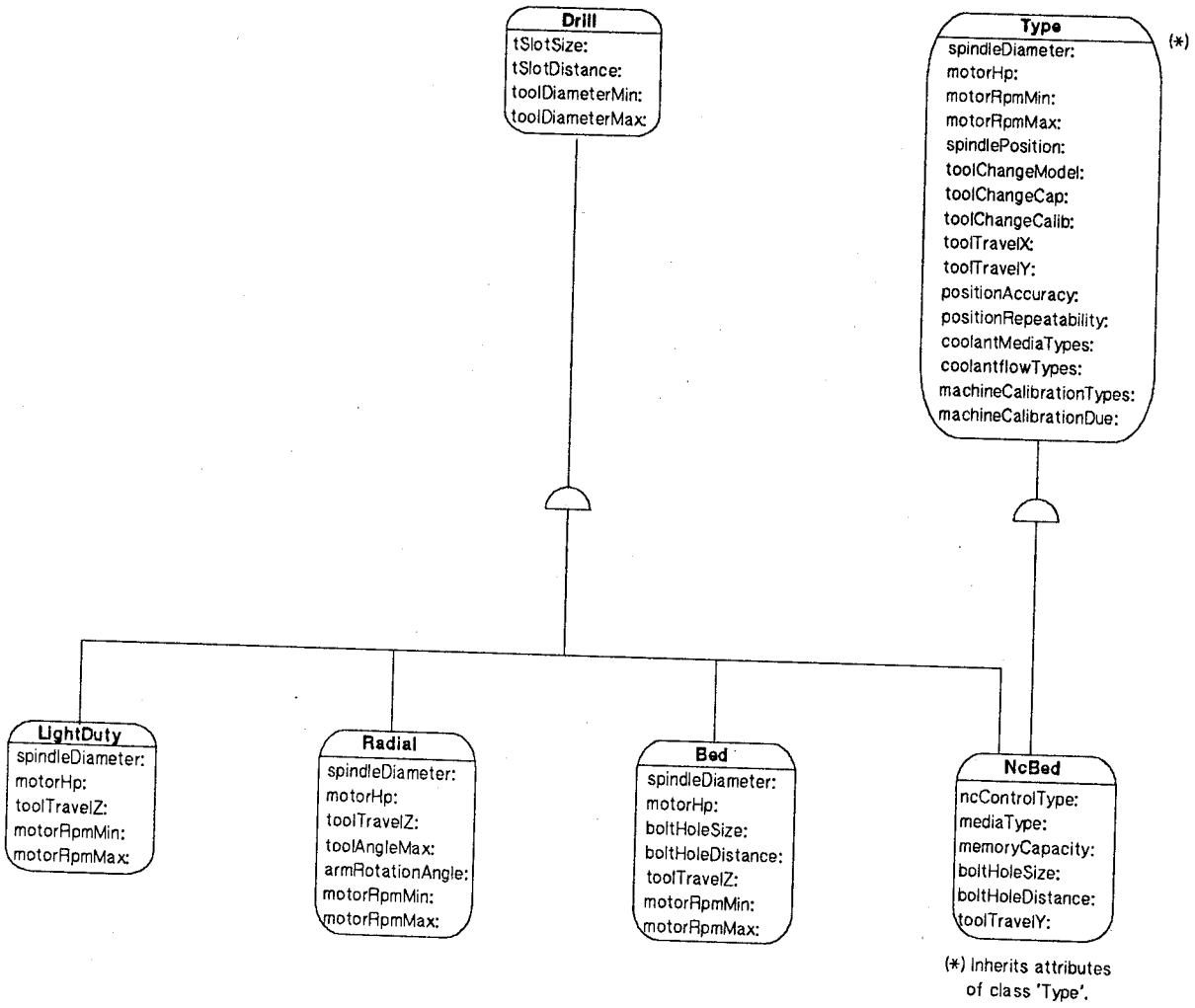


Figure B3: Machine Class Structure (continued)

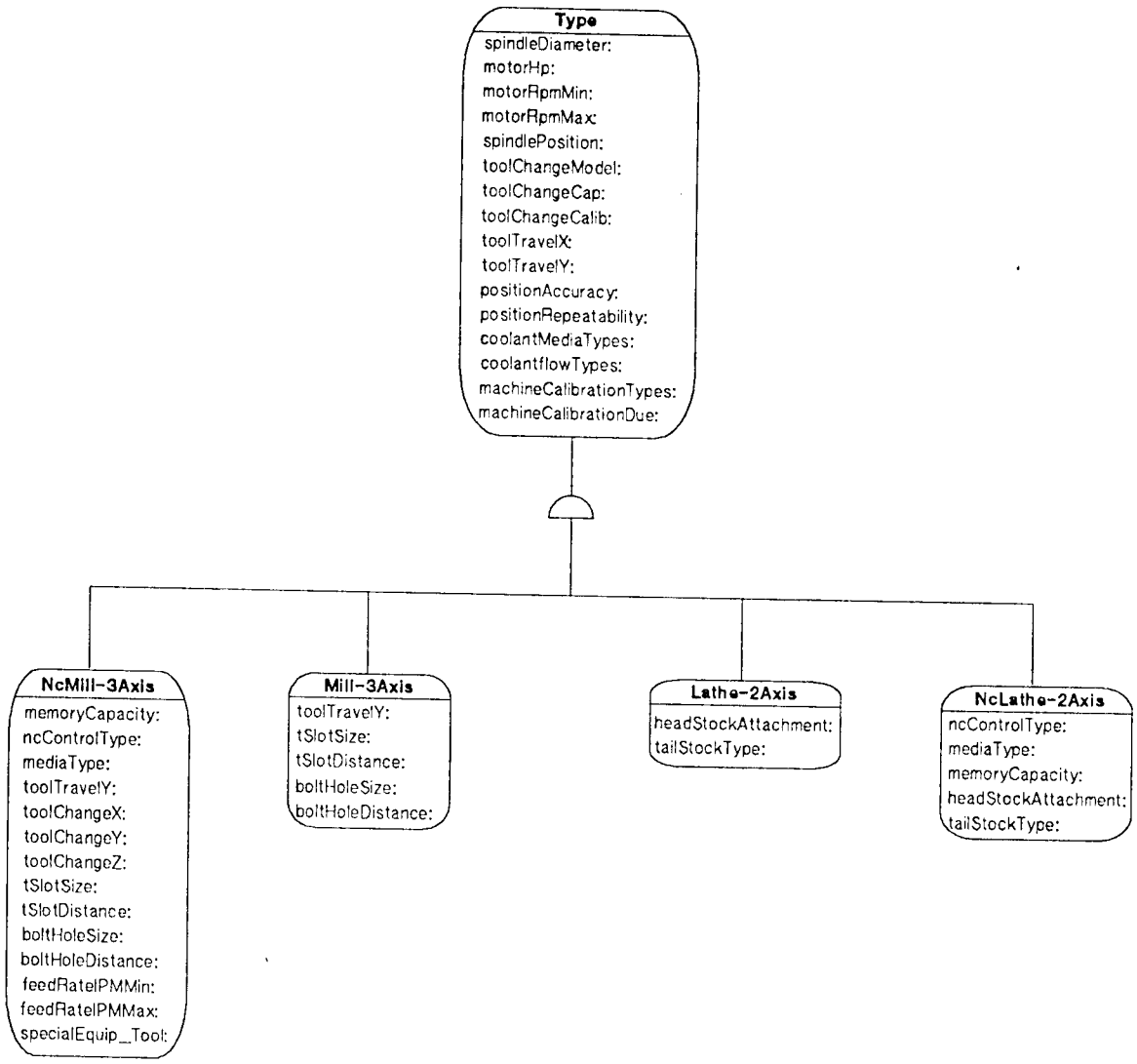


Figure B4: Machine Class Structure
(continued)

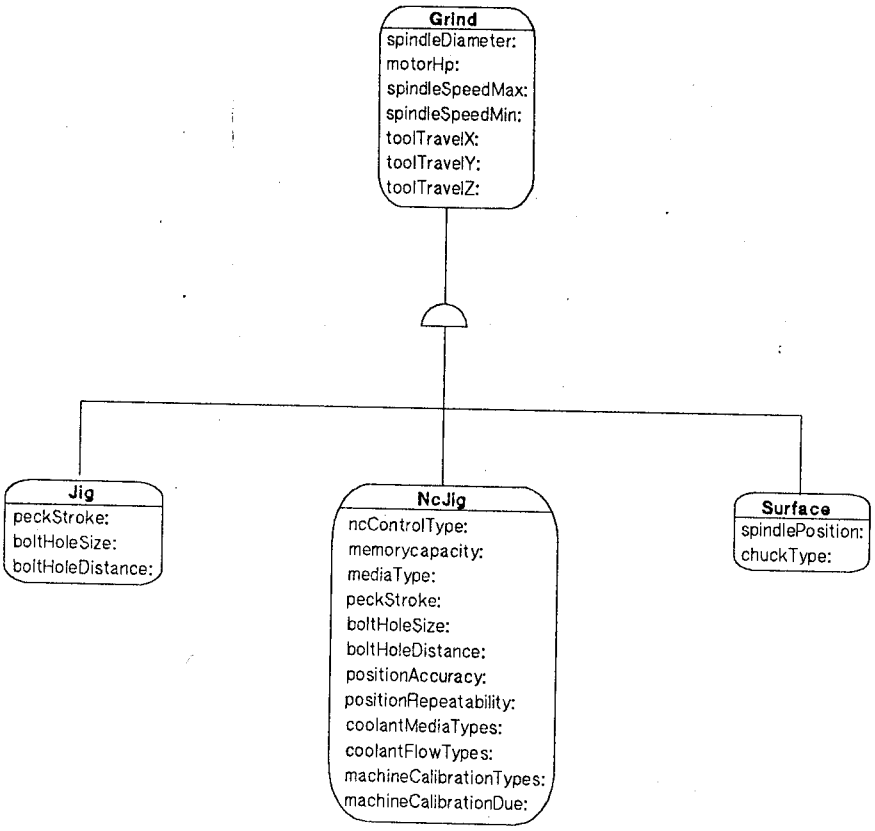


Figure B5: Machine Class Structure
(continued)

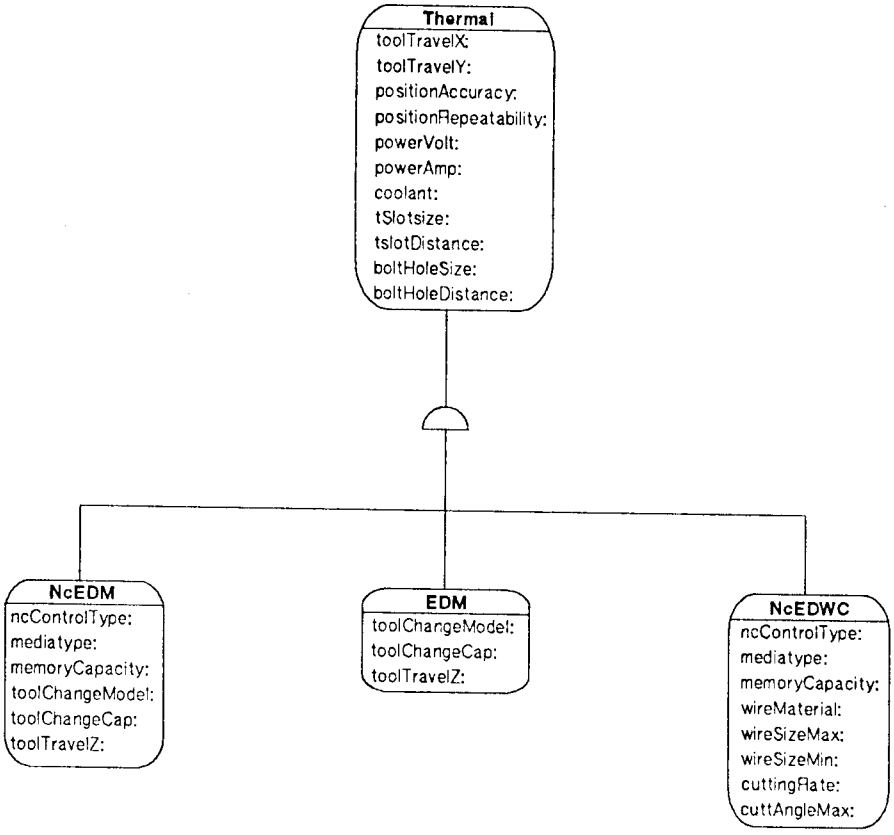
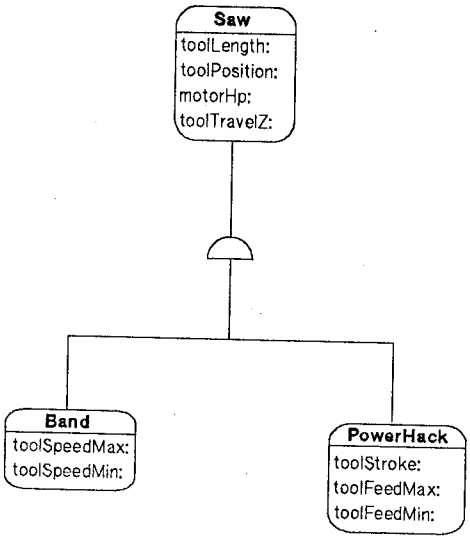


Figure B6: Machine Class Structure
(continued)



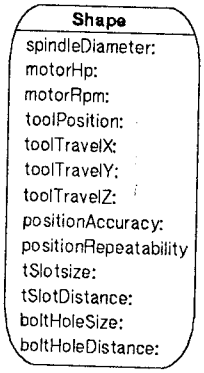


Figure B7: Machine Class Structure (continued)

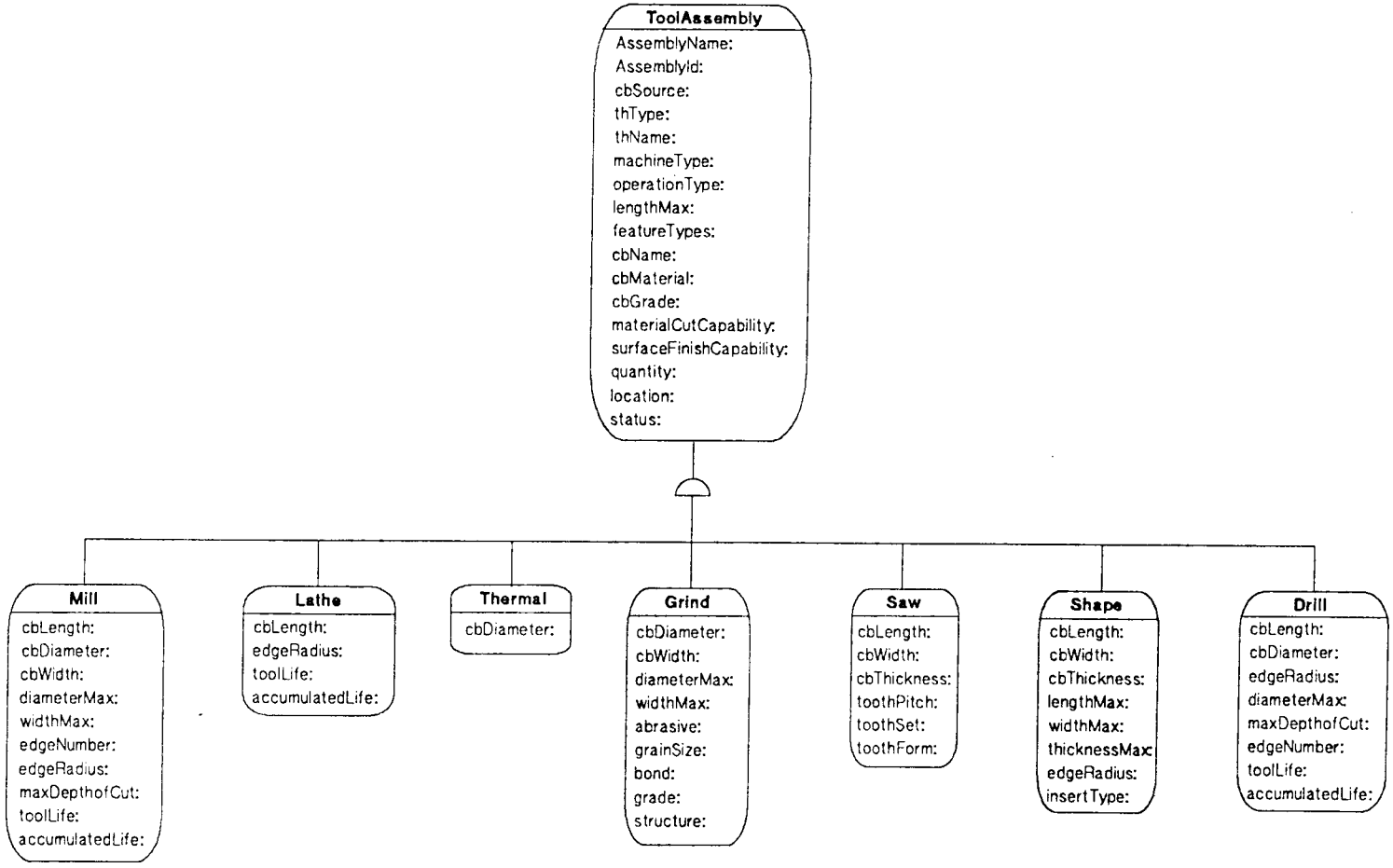


Figure B8: Tool Class Structure

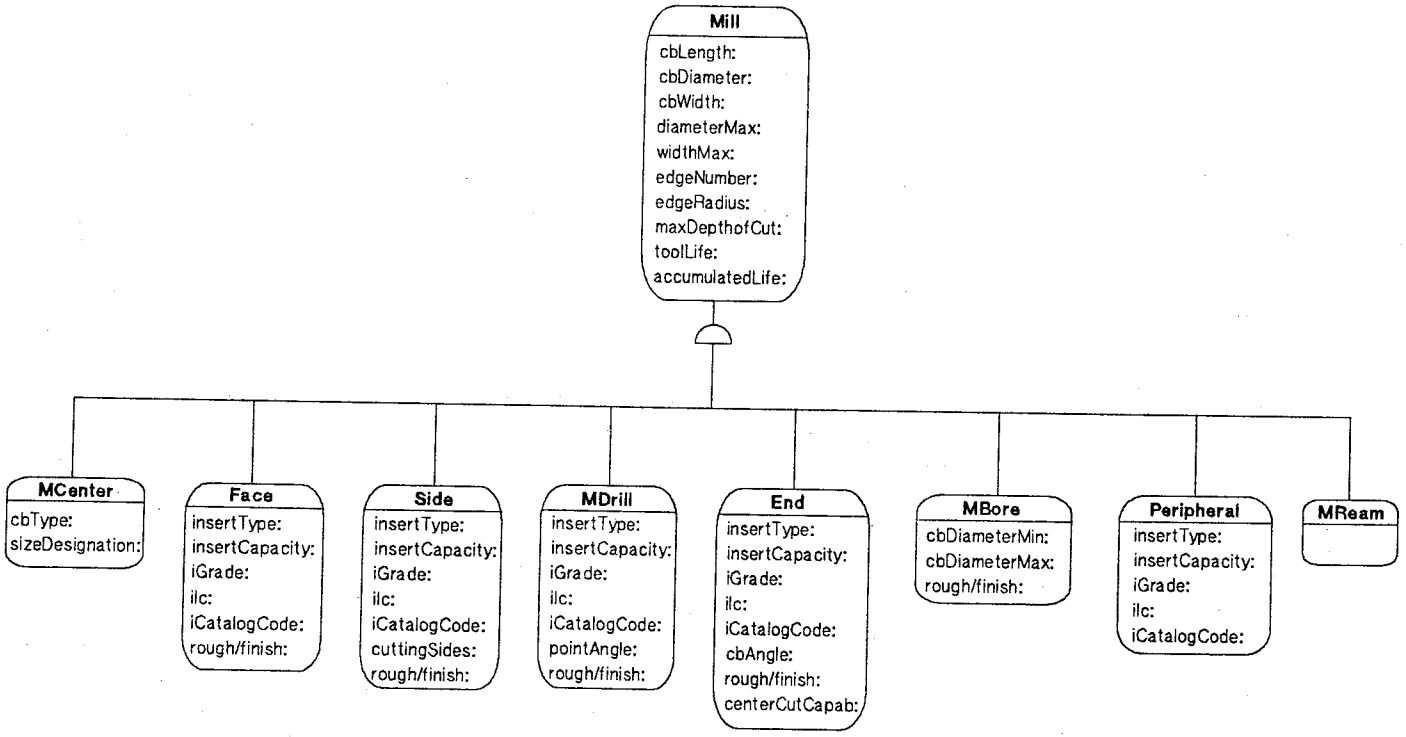


Figure B9: Tool Class Structure
(continued)

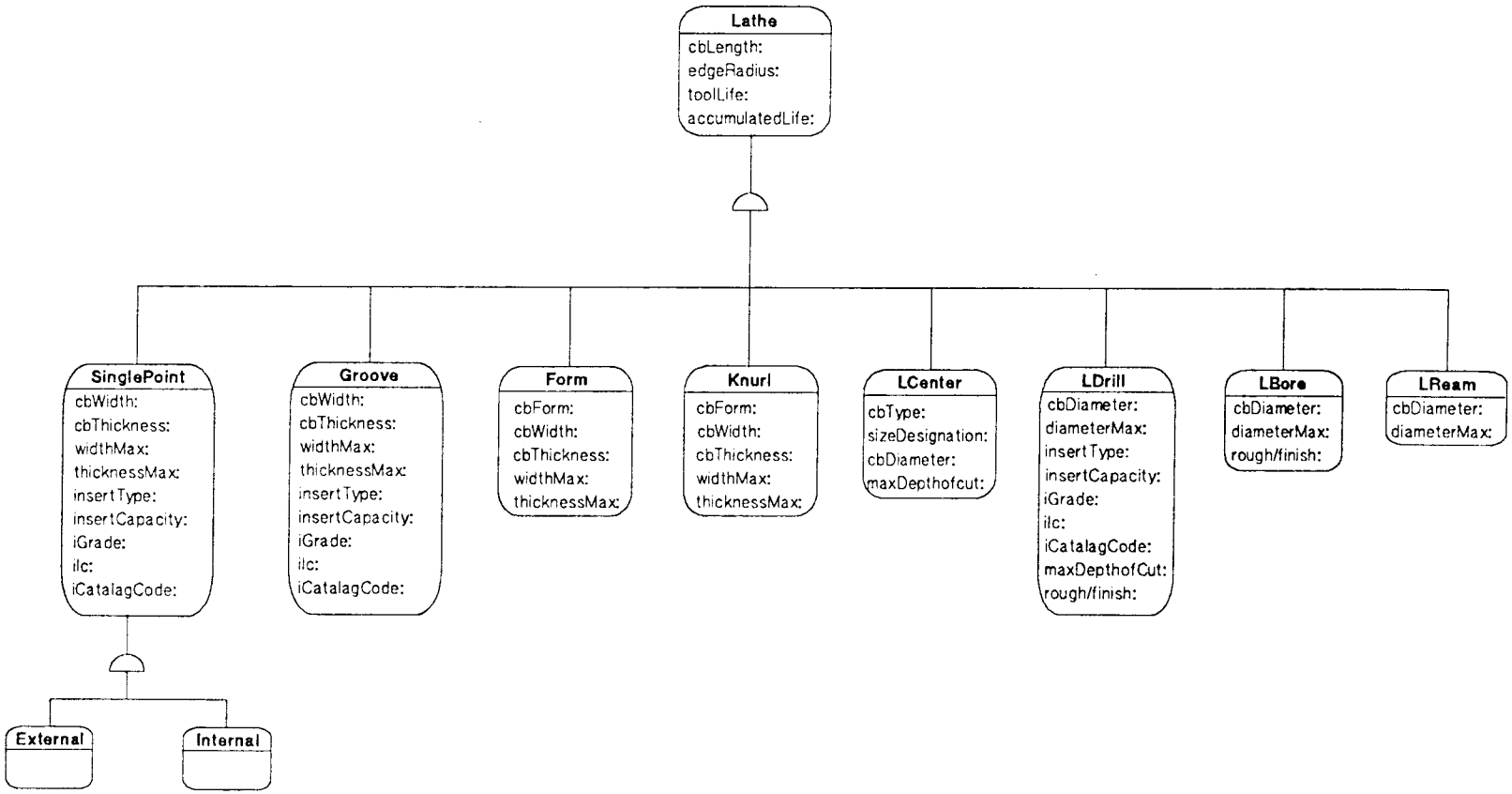


Figure B10: Tool Class Structure
(continued)

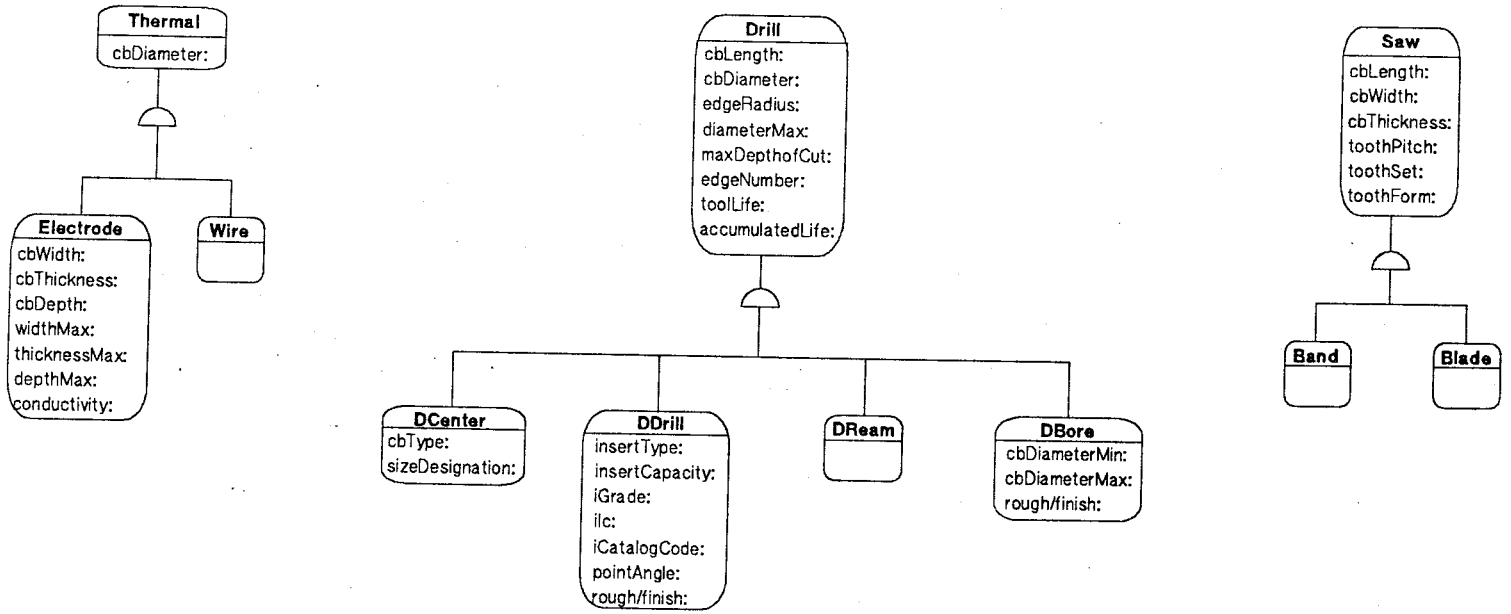


Figure B11: Tool Class Structure (continued)

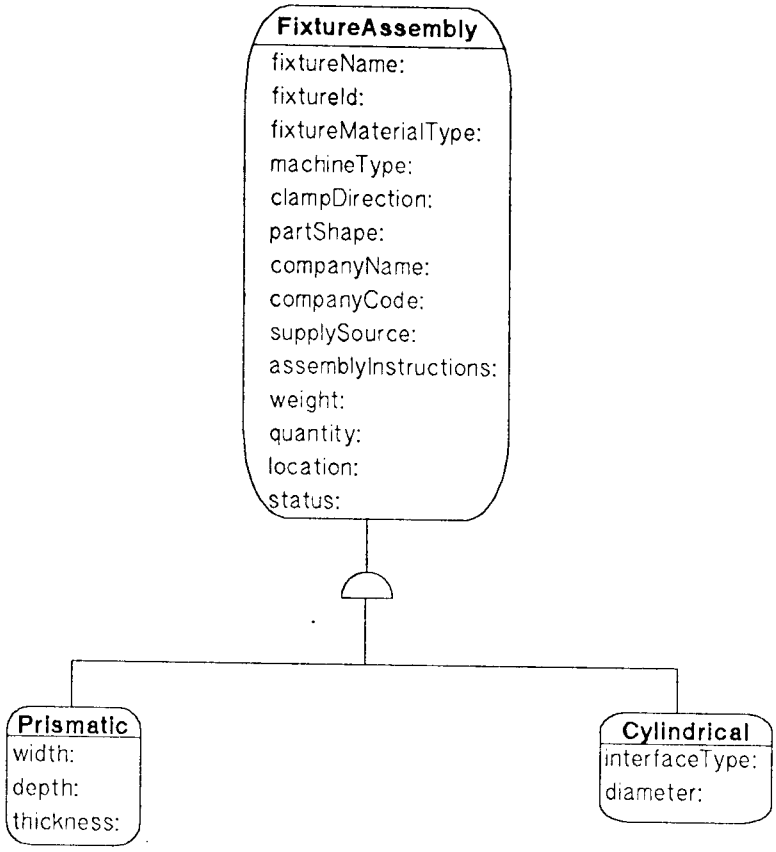


Figure B12: Fixture Class Structure

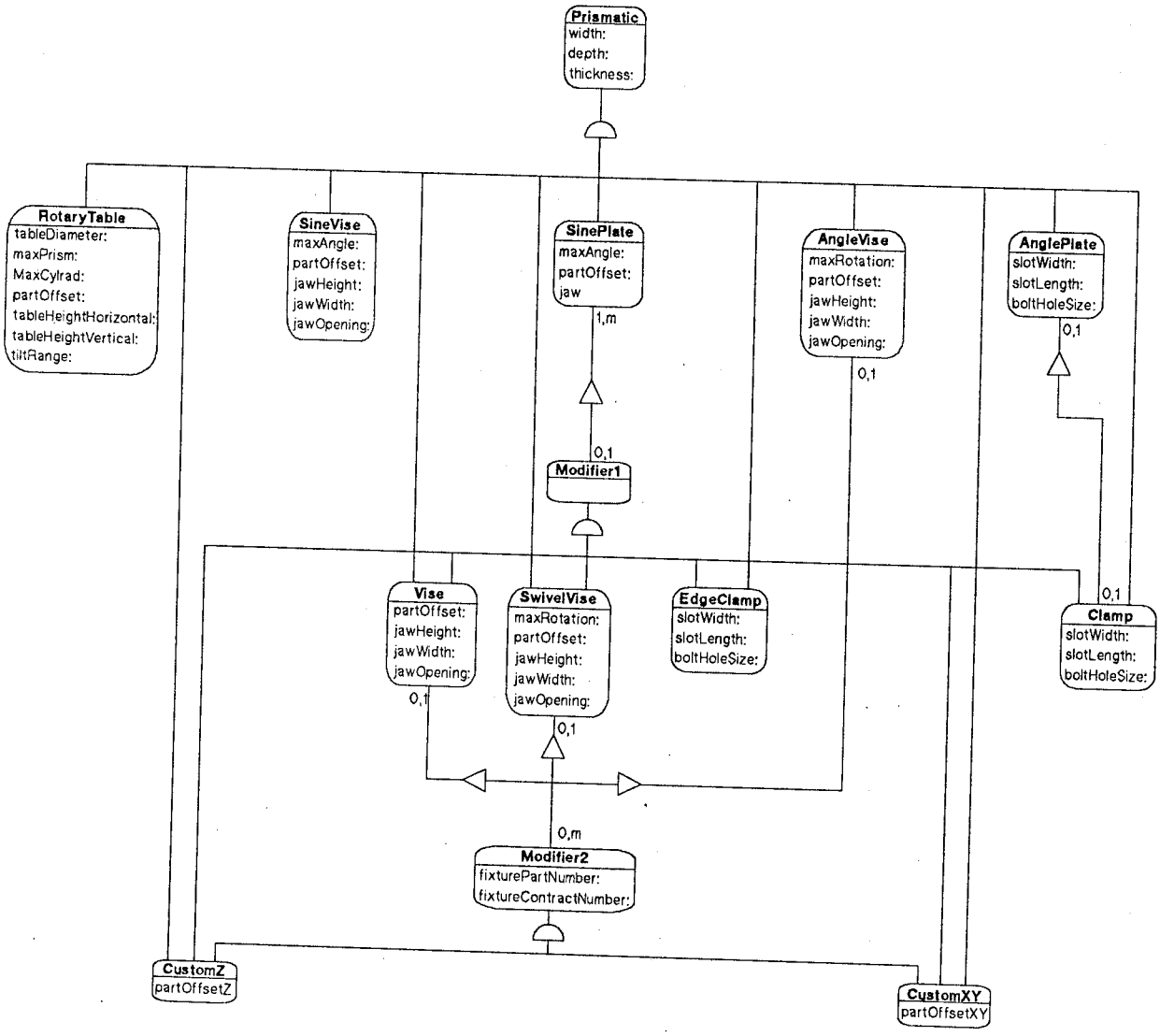


Figure B13: Fixture Class Structure (continued)

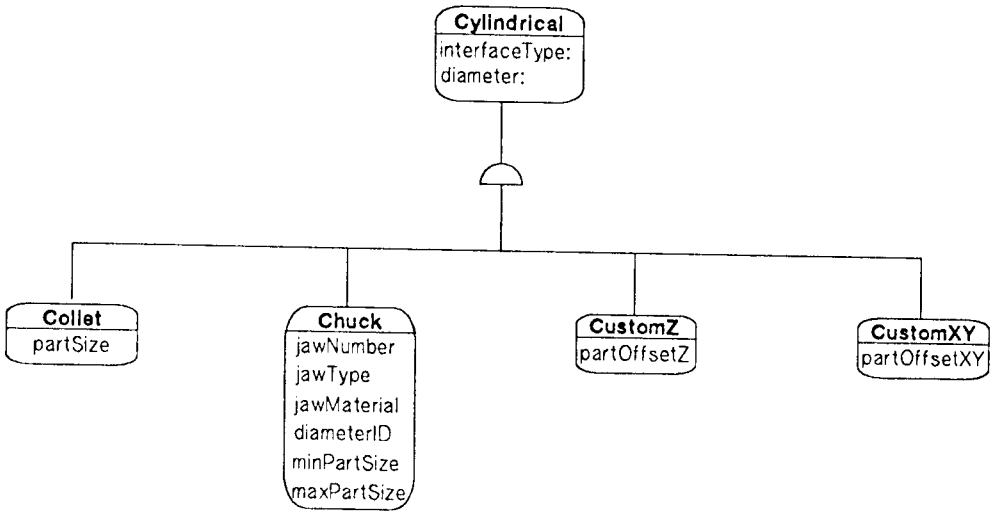


Figure B14: Fixture Class Structure
(continued)

Appendix C: Product Modeler

1 Modeling System Overview

The architecture of proposed feature modeling system is shown in Figure 2. It contains a product modeler, a feature editor, a feature extractor, a dimensioning/tolerancing modeler, and a feature rule manager. The product modeler invokes ACIS to create a product geometric model and to generate its corresponding feature model. An input to product modeler is a CSG tree. During geometric modeling, the CSG tree is evaluated and modeled in the B-rep form by the ACIS geometric modeler. At the same time, the feature shape description related to the primitive solid being used in the Boolean operation is attached to this B-rep model and feature graph is also constructed. Form features may be recognized at the modeling stages using the feature shape description on the B-rep. The feature extractor matches a piece of feature shape information with feature patterns in the feature rule library. It assigns a feature name to this piece of feature shape description if there is a feature rule matching this shape description. However, if a feature cannot be matched with all available feature rules in the feature library, the extractor will organize the feature shape description into a new feature rule and will prompt the designer to give it a name. This new rule is then added to the feature library by the feature rule manager for possible later use. In other words, the feature rule library grows with models are added. A new feature rule can be also input manually using the feature rule manager. The feature rule manager is also responsible for resolving possible conflicts and inconsistencies among feature rules in the feature library.

The feature editor provides an flexible way to modify a product at the feature level. The feature editor is different from a conventional geometric editor in that it not only carries out the geometric modification of a product but also handles the changes of its features, feature graph, and associated manufacturing attributes. To be able to efficiently attach manufacturing attributes such as datums, dimensional tolerances, and geometric tolerances to all geometric levels, the dimensioning/tolerancing modeler allows designer to click topological elements from a product on the screen and specify their desired manufacturing data. Finally, a graphical user interface is provided for easy access to the proposed feature modeling system.

2 Feature Representation

The proposed feature representation scheme defines a form feature by both its volume and surface. Each feature is also specified with three additional components: its net volume, the creating solid, and the Boolean operation. The surface and the volume are two versions of a feature description. The net volume is the volume added to or removed from the product model when creating a new feature. A creating solid is a solid used to create a form feature. A Boolean

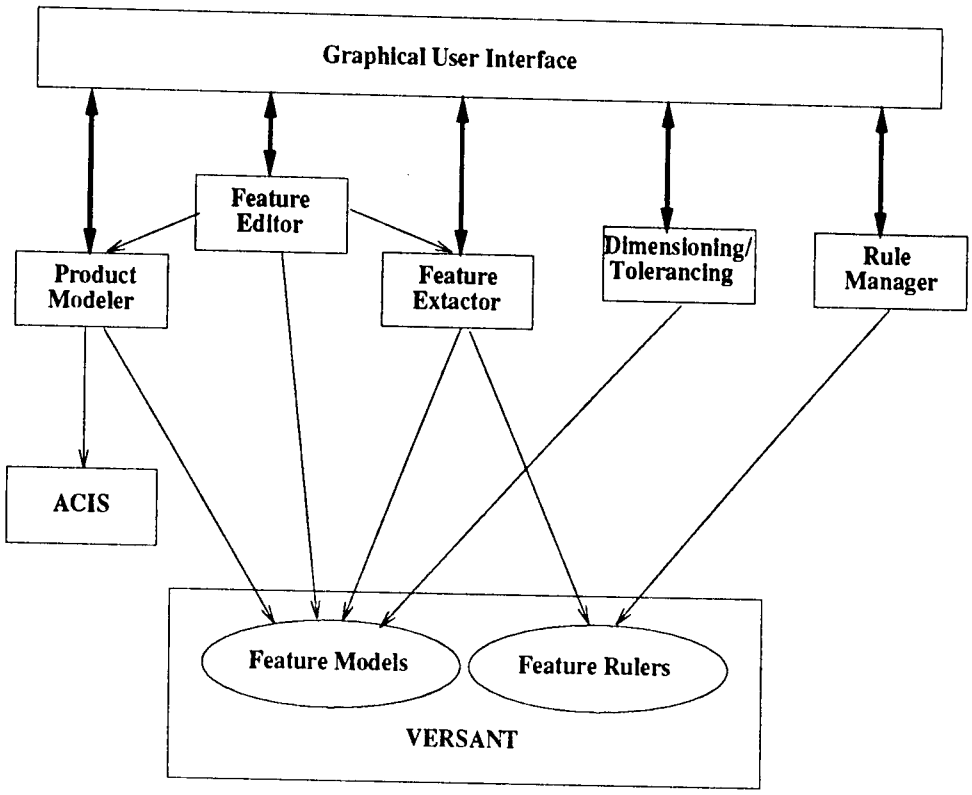


Figure 1: Architecture of Proposed Feature Modeling System

operation specifies a particular geometric operation. Only addition and subtraction operations are used to create form features for the proposed representation scheme. The creating solid, the Boolean operation, and the surface created by the operation are viewed as the major shape information of the form feature, and as such are critical to feature recognition.

Without loss of its generality, we assume each Boolean operation is valid. That is, a valid form feature is always created when a solid is added to or subtracted from a product model. In the proposed modeling approach, each Boolean operation modifies an existing product model in its B-rep form. For example, existing faces may be clipped or deleted from the product model, or new faces may be added to the model. New faces constitute the surface of the new feature. A 2-tuple indexing system is used to denote these faces and its creating solid. Each creating solid is assigned an ID when it is instanced. This ID is the first element of the index. The second index element denotes the ID of a physical face created by this modeling operation.

A feature volume is a solid instanced by a creating solid. Its size is evaluated from the minimum dimensions which enclose its corresponding feature surface. The size of a feature volume is changed when its surface is modified by a new feature. In other words, the feature volume reflects the actual dimensions of a solid primitive which encloses its feature surface. Each

feature volume is also specified with a position and orientation. In the proposed scheme, a feature volume is meant to support feature interaction checking and attribute attachment. The support for feature editing comes from the net volumes and the creating solids.

Due to the complexity of its shape, a net volume is represented in a B-rep form (in terms of two sets of bounding faces). One of the sets consists of the faces which are clipped from the product model by this operation. The other set consists of all new faces created on the product model. The same indexing system is used to name these faces. With a Boolean subtraction, a net volume is created by the intersection between the creating solid and the product model. With a Boolean addition, a net volume is created by subtracting the product model from the creating solid.

Net volumes are kept in the proposed scheme for two purposes. The first is to label each possible machine access direction. Under a Boolean subtraction, each clipped face is labeled; under a Boolean addition, each face added to the product model by this operation is labeled. The second purpose is to retain the result of each modeling operation in a modeling history. Net volumes are retained to support the handling of the geometry aspect of modeling operations, instead of feature operations. From this viewpoint, the geometry and topology of a net volume remain unchanged once created. Feature surfaces and feature volumes, on the other hand, may change as new features are added to the model.

The proposed presentation scheme captures each creating solid and Boolean operation in a CSG form to support feature editing and feature recognition. Seven solid primitives are defined for the proposed scheme. They are cuboid, wedge, cone, elliptical cone, cylinder, elliptical cylinder, and swept polyhedron. Each of these primitives is represented as a polyhedron in the product model. A wedge has four bounding faces; a cuboid has six. The number of faces of the other five solids, depending on their size, is determined as they are instanced. The bounding faces of each solid type are shown in Figure 3 and identified with a face ID. The shape information of a feature is directly derived from the shape of its creating solid. During a Boolean operation (either addition or subtraction), the faces of a creating solid are separated into two face sets by the intersection between the solid and the product model. One face set contains the faces on the product model which jointly define the feature surface. The topology of the feature surface is determined by the creating solid. The topological pattern of the face set provides unique information for feature recognition.

Figure 3 summarizes the five feature components as described in the above paragraphs. Figure 4 shows a form feature called **blind_step**, which intersects an existing slot on the prismatic part design. This form feature is created by subtracting a cuboid solid from the then current product model. The surface of the feature consists of three interconnected faces. The net volume of the feature is the actual volume removed from the product model by the operation. The feature

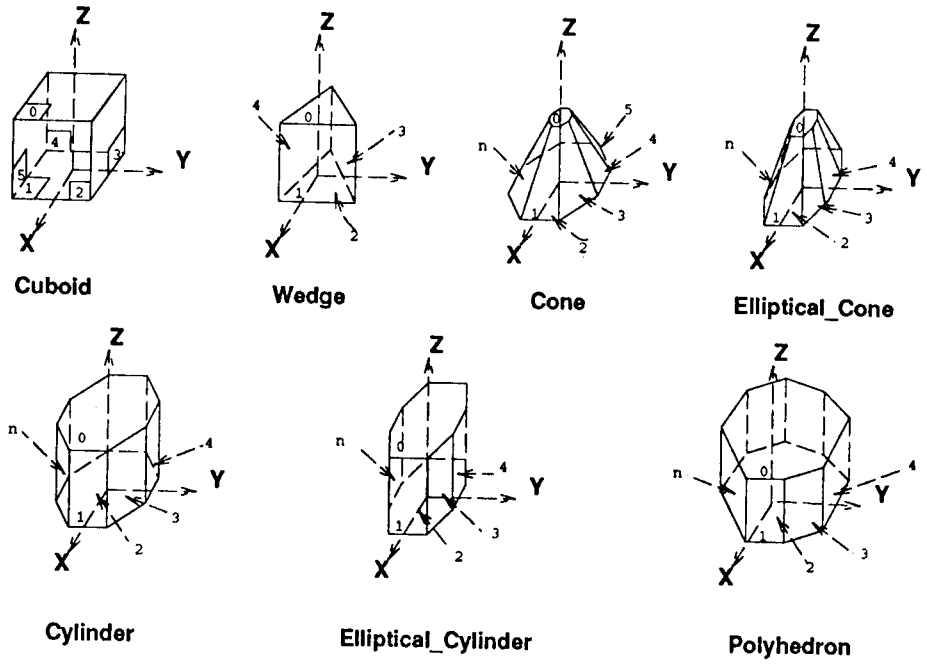


Figure 2: Solid Primitive Definition

volume takes the shape of the creating solid, its size being calculated as the minimum dimensions required to enclose all the faces on the feature surface. Note that the feature surface may change its geometry and topology. While the feature volume may change its dimensions, the net volume and the creating solid remain unchanged.

3 Feature Graph Representation

A feature graph is used to capture the feature adjacency relationship and the modeling history of a product design. Like a typical feature graph, the proposed graph consists of nodes and directed arcs. Each node represents a feature. Each arc links a pair of intersecting features, from one created earlier to the one created later. Each arc is also designated as active or inactive in order to further distinguish between the features which are currently adjacent and those which used to be adjacent. This feature graph is constructed as the geometric modeling proceeds. This feature graph and individual features continuously evolve as new features are added to the graph.

After each Boolean operation, all five elements of the new form feature are constructed. The adjacency relationship between this feature and other features is established in three steps: 1) all faces adjacent to a face of this new feature are labeled, 2) each solid ID corresponding to a labeled face is then identified, and 3) all features corresponding to the solids identified in the second step, except the creating solid itself, are recognized as the adjacent features. Once an

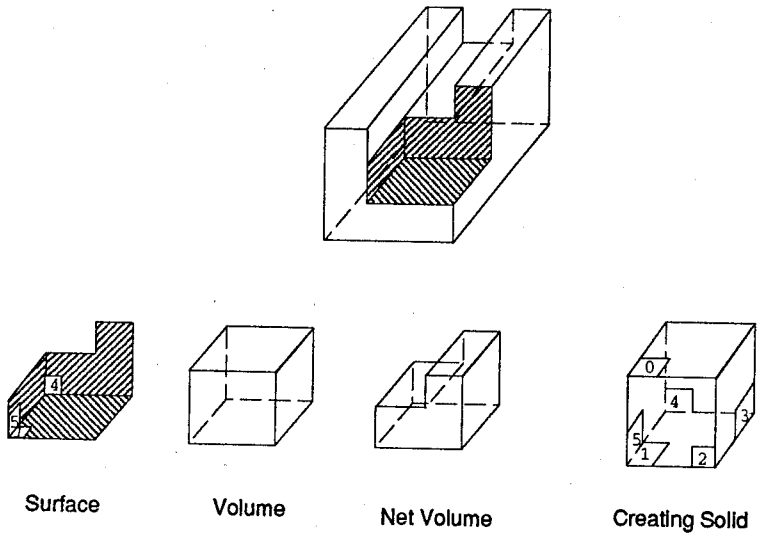


Figure 3: A **blind_step** Feature and its Components

adjacency relationship is found, an arc is added to its adjacent feature, pointing to the new feature.

Additional changes to the feature graph and existing features may occur with the addition of a new feature. Each new feature brings a change to the bounding faces of the product model. The faces of a feature may be modified without changing its feature type, but any deletion of its faces may cause a change in its feature type. A modification to a face (without deleting it) does not cause any change to any face index of the feature surface but may change the dimensions of its feature volume. When adding a new feature, the volume of each existing feature may remain the same or decrease. When a feature type is changed due to a feature addition, the indexes of its feature surface are adjusted by deleting the ones whose corresponding faces are removed from the product model. While the feature volume type always remains unchanged, its size is re-evaluated to just enclose the modified faces on the feature surface.

The existing adjacency relationship in a feature graph may also be altered when adding a new feature. Figure 5 shows a sequence of changes to the feature graph due to the addition of new features. Particularly note that the addition of **Feature3** deactivates the adjacency relationship between **Feature1** and **Feature2**. The solid arcs denote an active adjacency relationship, and the dashed arc denotes an inactive adjacency relationship. With the use of inactive arcs, the modeling history is maintained by capturing all deactivated adjacency relationships.

The deactivated relationships are needed for feature editing because, in order to delete a form feature, all of its descending nodes must first be undone and then sequentially redone after the feature is deleted. The undo and redo steps are identified and sequenced by reversing each related

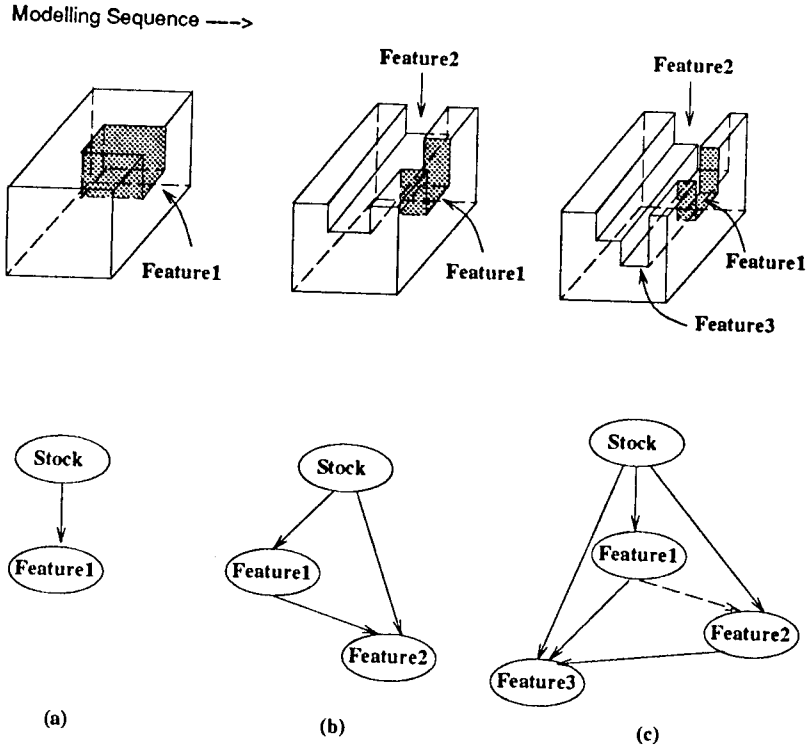


Figure 4: Construction of a Feature Graph

arc (directly or indirectly), starting from the affected node at the bottom of the graph. Using the adjacency relationship defined in Figure 5(c), the deletion of **Feature1** can be correctly carried out in the following steps: 1) undo **Feature3**, 2) undo **Feature2**, 3) undo **Feature1**, 4) redo **Feature2**, and then 5) redo **Feature 3**.

Without inactive arcs, feature editing may lead to an unexpected result. Figure 6 shows an example of this problem. Based on the same feature graph as presented in Figure 5(c) but without the inactive arc, the procedure for the deletion of **Feature1** becomes: 1) undo **Feature3**, 2) undo **Feature1**, and 3) redo **Feature 3**. In this procedure, **Feature2** is not undone and redone because its adjacency relationship with **Feature1** no longer exists. As a result, the portion of the **Feature1**'s volume which is enclosed in the **Feature2**'s volume is erroneously added back to the product model as shown in Figure 6(c).

Adding a new feature may also cause deletion of an existing feature from the product model. In order to support feature editing, the node of the deleted feature is retained in the feature graph. Its feature surface and feature volume are both nullified but the other three elements are kept intact. All arcs connected to the nullified node are changed into inactive.

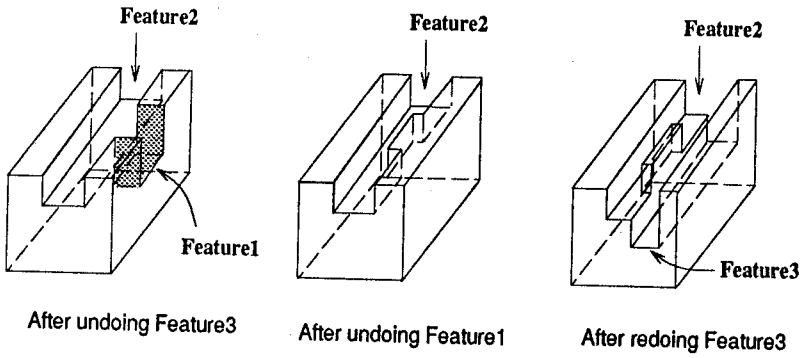


Figure 5: A Feature Editing Problem

4 Feature Recognition

This section discusses a feature recognition approach supported by the proposed representation scheme and the management of feature rules.

4.1 Feature Recognition Approach

The proposed feature recognition approach is based on the assumption that each Boolean operation creates a form feature and each form feature can be defined by the bounding faces of a primitive solid. Under this assumption, feature faces can be collected at modeling stages. They can be either recognized at modeling stages or recognized after modeling stages. Meanwhile, feature patterns are defined with a subset of faces on the creating solid according to the shape of pre-defined primitive solids. For example, any two contiguous bounding faces of a cuboid solid define a **step** feature.

Two formats of feature recognition rules are developed. One format is for cuboid and wedge solids whose bounding faces are known before they are instanced. The other is for solids whose bounding faces are determined when they are instanced. The other five solid primitives as defined above fall in this category. In addition to face patterns, recognition rules also make use of the information of solid types and Boolean operations. The first format of feature recognition rules takes the following form:

IF *solid_type* & *face_pattern* & *Boolean_op* **THEN** *feature_type*

where *solid_type* is either a cuboid or a wedge; *face_pattern* is a subset of faces bounding a cuboid or a wedge; and *Boolean_op* is either a Boolean addition or a subtraction. Applying this format of rules to the feature with hatched faces in Figure 4, a feature rule that recognizes its feature

type as *blind_step* can be explicitly written as:

IF *"Cuboid" & {1, 4, 5} & "-"* **THEN** *blind_step*

The second format of feature recognition rules accepts the solids whose bounding faces are undetermined until they are instanced. The solids in this format commonly possess a top face and a bottom face. As shown in Figure 3, the top face is marked as the face 0 and the bottom is marked as the face 1. Thus the only variation is in the number of side faces. The ratio of these side faces on the product model and the total number of side faces are calculated and used to partially determine the major shape of the form feature. This ratio ranges from 0 to 1.0. A ratio of 1.0 means all side faces are on the product. With the information on its top and bottom faces, a feature can be deduced as following: if neither the top face nor the bottom face is on the product, the operation creates a hole-like form feature, if the ratio is 1; otherwise, a slot-like or a step-like feature is created, depending on the ratio. If the top face or the bottom face is on the product model, the modeling operation created a pocket, a blind_slot, or a blind_step feature. This format of feature rules can be generalized as:

IF *solid_type & top_face & bottom_face & face_ratio & Boolean_op* **THEN** *feature_type*

where *solid_type* and *Boolean_op* have the same meaning as defined in the first format. *Top_face* and *bottom_face* take a binary value of 1 or 0, indicating the existence of the top face and the bottom face on the product. Four sample rules having this format are listed. Their corresponding features are shown in Figure 7(a) to 7(d) with a cylinder solid and a Boolean subtraction, where ϵ is a variation allowed for a face ratio.

IF *"Cylinder" & 0 & 0 & 1.0*$\pm\epsilon$ *&* *"-"* **THEN** *cylindrical_through_hole*
IF *"Cylinder" & 0 & 0 & 0.5*$\pm\epsilon$ *&* *"-"* **THEN** *cylindrical_through_slot*
IF *"Cylinder" & 0 & 0 & 0.25*$\pm\epsilon$ *&* *"-"* **THEN** *cylindrical_through_step*
IF *"Cylinder" & 1 & 0 & 0.5*$\pm\epsilon$ *&* *"-"* **THEN** *cylindrical_blind_slot*

Feature recognition rules can be generated during a modeling process or prepared manually. While the recognition of form features can be either carried out at each modeling operation or delayed until the design is completed, all feature related data are collected at each modeling operation. Each feature component is calculated and organized into a premise according to one of the above two formats. If the condition is matched with an existing feature rule, a feature is said to be recognized. If the condition does not match any existing rule, a request is generated to

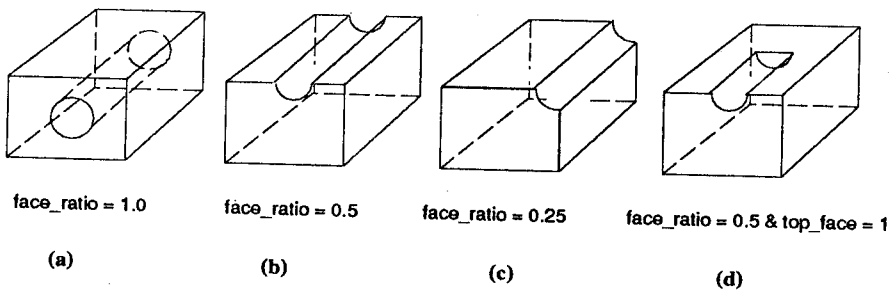


Figure 6: Various Features Created by a Cylinder

name this feature and to convert the condition into a new rule. There are two ways to associate a feature type to a feature rule premise. One way is to impose the naming responsibility on the designer, who provides a feature name for each resulting form feature premise presented to him by the system. Another way is to update the system by matching feature surfaces with pre-defined feature patterns. These two approaches complement each other and should both be implemented in a modeling system. The first approach ensures the acceptance of a newly created feature rule but may lead to the inconsistent naming of form features. The second approach consistently assigns a feature name to a feature recognition rule but its capability is limited by pre-defined feature patterns.

A complete feature graph must be maintained at all times in order to recognize form features during the modeling process. Each time a new feature is created on the product model, its adjacent features on the product design may be modified. A feature extractor, therefore, is needed to check and modify the feature type of each existing feature during the modeling step. This approach permits concurrent product and process design and on-line design evaluations. The modeling process may, however, be less efficient. One way to improve the efficiency is to delay feature recognition until modeling has been completed. This results in no repetitive feature editing or modifications.

4.2 Feature Rule Management

In order to facilitate the correct addition of feature rules to the feature library, a feature rule manager is developed to allow the designer to insert, delete, or modify feature rules manually. It is also responsible for the resolution of possible conflicts, inconsistencies, and redundancies that may occur between feature rules because of manually input.

Conflicts occur when the same premises have been used to match different feature types. It happens due to a failure in realizing that the same premises have already been used to match

another feature type. Redundancies mean that the same rules have been defined in the feature library more than once. An inconsistent rule indicates an mismatch between premises and their feature, for example, when three contiguous faces have been used to define a feature **step** which consists of two perpendicular faces.

Conflicts and redundancies can be avoided in the feature library by checking a newly-added feature rule with existing rules. To check consistence in a feature rule, two meta-rules are built to filter a new added feature rule. The first meta-rule is the number of logical faces for a feature pattern. For instance, if a key word **step** appears in a feature name, this rule must be contains two faces. If the key word **step** is appeares accompanied by the key word **blind**, this rule must have four faces. If this meta-rule can not fit the rule to be added, the rule manager will ask the designer to confirm the correctness of this new rule before storing it. The second meta-rule is that a feature rule has to be defined with contiguous faces. Through this meta-rule, each time a new feature rule is to be added, the rule manager will check its feature surface to ensure that all these it feature faces are contacted each other.

5 Dimensions, Tolerances, and Datums

Dimensions, tolerances, and datums are the major manufacturing attributes assigned to a product. These attributes are essential to manufacturing. A comprehensive discussion on these topics can be found in Fleming [6]. In this development, dimension, tolerance, and datum representation is studied and implemented in the proposed feature modeling system.

5.1 Representation Requirements

Dimensions and Tolerances include specifications of *size, form, orientation*, and *position*. Size tolerance is the dimension allowance of a geometric entity. Form tolerances indicates a surface variation limitation such as plane flatness, edge straightness, cylinder circularity. Orientation tolerance includes parallelism, perpendicularity, and angularity of a geometric entity. Finally, position tolerance is the distance measurement between two entities. Other tolerances are called geometric tolerance, except size tolerance which is usually called dimensional tolerance, The representation of dimensional tolerances is only concerned with its feature volume, while geometric tolerances may concern topological elements at all geometric levels. For example, a straightness tolerance needs to be specified with respect to an edge, a flatness tolerance attached to a face, while a position tolerance should be associated with a feature volume and a feature face if the distance between a cylinder and a face needs to be specified. Both geometric tolerances and dimensional tolerances can be captured in individual features because all topological elements in

a product belong to one or more features.

In addition to geometric and dimensional tolerances, a datum system must be be used as the standard reference to geometric tolerances in a product. A datum may be a surface, a line, and a point. It should be captured in the product node of a feature graph so that it is accessible to all geometric tolerances.

5.2 Representation Scheme

The representation of geometric tolerances, dimensional tolerances, and the datum system are independent of each other in the proposed feature modeling system. The date structure for geometric tolerances is defined such that it includes tolerance type, tolerance location, referenced object, and a cross reference pointer to its possible partner. The tolerance type is a geometric tolerance name such as "flatness", "straightness", and "parallelism". The tolerance type also implies whether one or more referenced object are required to specify this tolerance. For example, a flatness tolerance does not need a referenced object, while a parallelism tolerance may need a datum as its referenced object. The tolerance location is the address of either a feature volume, a feature face, or an edge. Both a volume and a face could be defined by the 2-tuple ID naming mechanism introduced in Chapter 4. To represent a volume, simply have face ID be -1. To represent an edge, in addition to the feature face with this edge, its location needs to be further denoted by the loop ID on this feature face and its edge ID on this loop. Datums can also be represented in this way because they are located on one of the topological elements of a product. Dimensional tolerances can be represented by attaching its value to their corresponding features directly. Each feature can hold one dimensional tolerance. However, the representation formats for a dimensional tolerance may vary with the type of primitive solids. For instance, a dimensional tolerance for a block has a format (depth, width, height), while for a cylinder the format is (radius, height).

Figure 8 illustrates the representation of two datums, one dimensional tolerance, and one position tolerance in a simple product including two features **holes**, and one feature **step**. As shown in Figure 8, datums A, B are captured in product node. Both of these datums have type "plane" and located at face (0,1) (instance ID, face ID) and face (0,2) (instance ID, face ID) of base_part. The position tolerance is captured with two copies in both **holes**. These two copies are kept consistently by their cross reference pointers in the data structure. The locations for position tolerances are denoted in the figure by (1,-1) and (2,-1), respectively, because two feature volumes are involved. Finally, the dimensional tolerance for feature **step** is denoted with depth, width, and height (because its feature volume is a block). The depth value 0.05 is a default value because it is not given by the designer.

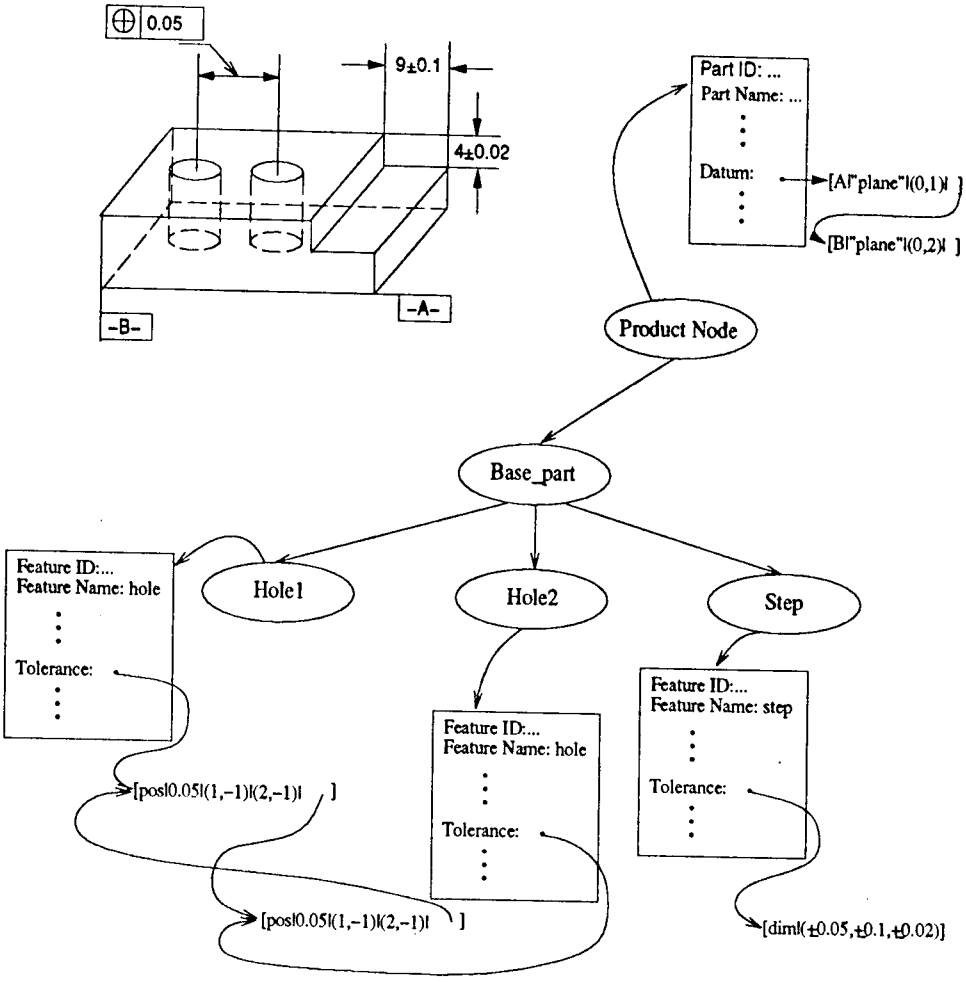


Figure 7: An Example of Dimensions, Tolerances, and Datum Attachment

5.3 Validity Checking

The validity checking for datums, dimensional and geometric tolerances includes the detection of tolerance redundancies and tolerance violations. Redundancy checking detects whether one tolerance or datum has been specified twice. A datum location corresponds to a topological element of a product. This location can only be used as a datum once. Therefore, it is easy to check if the location of a new datum has been used or not. The repeated use of a topological element as a datum will be recognized as a redundant datum. For dimensional tolerances, redundancies means that a feature has been specified more than one dimensional tolerance. In this case, the dimensioning/tolerancing modeler will ask designer to either chose one of them or modify an existing one. To check if a geometric tolerance has been redundantly specified to a product, tolerance type, tolerance location, and even the referenced object of this geometric tolerance should be compared with existing geometric tolerances. These components are required

to determine the uniqueness of a geometric tolerance. Similarly, a redundant geometric tolerance is also handled by asking designer to either chose one of them or modify existing one by the dimensioning/tolerancing modeler.

Tolerance violations indicate conflicts of geometric tolerance specifications in a product. They can be further classified into physical violations and semantic violations. A physical violation implies that a tolerance has been assigned to an unexisting topological element, an unexisting topological element has been used as a referenced object, or both. It is simple to examine whether or not a tolerance to be assigned suffers from a physical violation. This is done by checking its location and its referenced object if it has one.

A semantic violation indicates a contradictory attachment of two geometric tolerances to a product. It happens to a geometric tolerance whose specification needs to be completed with one or more referenced objects. At present time, the proposed feature modeling system supports ten types of geometric tolerance. They are: 1) flatness; 2) finish; 3) straightness; 4) axis straightness; 5) circularity; 6) cylindricity; 7) angularity; 8) perpendicularity; 9) parallelism; and 10) position. Among these geometric tolerances, types 1) - 3) are self-referenced tolerances (therefore, there is no semantic violation happen to them), the remains involve one or more referenced objects, either datums or other topological elements in the product model. In regards to these types of geometric tolerances, the dimensioning/tolerancing modeler is able to do the following semantic violation checkings: 1) perpendicularity and parallelism assigned on the same location cannot have the same referenced object; 2) the location of a flatness or a finish tolerance can only be a face; 3) straightness can only be located on an edge; 4) axis straightness, circularity, and cylindricity tolerances have to be assigned to a cylinder.

6 Feature Editing

The proposed feature modeling system provides editing functionalities at feature level with following operations: *undo*, *redo*, *insert*, *delete*, *modify*, and *replace*. As we have mentioned in Chapter 5, the proposed feature graph keeps track of the feature modeling history of a product in its life cycle with both active and inactive arcs. This feature modeling history makes it possible to carry out all feature editing operations at the feature level. Feature editing operations not only changes the geometry of a product but also may affect the correctness of the associated manufacturing attributes. Therefore, corresponding adjustments to the manufacturing attributes are needed to keep their manufacturing meaning.

6.1 Feature Editing Operations

Among the six feature editing operations listed, *undo* and *redo* are two most basic ones. An *undo* operation simply undoes the last Boolean operation by removing corresponding feature node and related arcs from the feature graph. This node, together with its Boolean operation, is then pushed into a system-maintained solid stack for later possible "redo" operation. A *redo* operation invokes up the primitive solid from the top of solid stack if it is not empty and then carries out its associated Boolean operation. The result of a "redo" operation will add a new feature to the feature graph. Notice that "redo" a primitive solid may not recover the product to its original state when this primitive solid is "undone" if it is not "redone" immediately after corresponded "undo" operation. It is possible that many features could have been added or deleted between the "undo" and "redo" operations. The "undo" operation is implemented by re-inserting the net volume of the undone feature into to the product model, while the "redo" operation is implemented on the creating solid of the undone feature in order to maintain a correct feature interaction relationship with other features in the product.

An *insert* operation add a new feature into a product by executing a Boolean operation on a new solid instance. The behavior of an *insert* operation is the same as a *redo* operation except that it obtains the solid instance from the designer instead of solid stack. A *delete* operation allows the designer to remove a feature other than base part from a product. It is implemented with a sequence of *undo* and *redo* operations. To finish a delete operation, all descending nodes of the feature to be deleted have to be undone first. The feature is then removed from the feature graph but not put back into the solid stack (therefore, it cannot be redone later). Finally, all feature nodes originally being undone are redone in a reverse sequence. A delete operation is discussed in Chapter 5.

A *modify* operation allows the designer to change the dimensions of a desired feature in the feature graph. It also allows the designer to translate and rotate the feature. The implementation of a *modify* operation is the same as the *delete* operation except that the feature to be deleted will be redone after its modification with the *modify* operation rather then being discarded through the *delete* operation. Finally, a *replace* operation lets the designer modify a feature by changing its creating solid and its associated Boolean operation. It will replace the original feature with a new feature in the same position as the original one's. The *replace* operation acts as in a similar way by carring out a *delete* operation and then an *insert* operation. Unlike an *insert* operation which will put the newly-added feature at the end of modeling history, a *replace* operation will keep the original modeling history. The *replace* operation is implemented in a way similar to that of a *modify* operation.

6.2 Manufacturing Attribute Adjustment

Each of above feature editing operations may modify a feature graph, feature geometry, feature adjacency relationships, and feature manufacturing attributes. The modification of feature geometry and feature adjacency relationships have been discussed in Chapter 5. The possible modification of associated feature manufacturing attributes due to feature editing operations is now examined.

The feature manufacturing attributes, as defined in the proposed feature modeling system, include geometric tolerances, dimensional tolerances, and datum system. Because all these attributes are located at some topological element of a product, modification to the geometry of this product will inevitablely affect the specification of these attributes. The following table summarizes the possible interaction relationships between feature editing operations and manufacturing attributes. The cross mark inside a box indicates that the attribute in the corresponding row may be modified by the operation in the corresponding column.

	undo	redo	insert	delete	modify	replace
Datum	X	X	X	X		X
Dimensional Tolerance		X	X	X	X	X
Geometric Tolerance	X	X	X	X	X	X

In the case of a datum, because of its physical association with a topological element of the product, any possible deletion of topological elements from the product may destroy is. *Undo*, *delete*, and *replace* are operations that may eliminate topological elements directly from a product. An *insert* operation may indirectly delete a datum if a newly added feature overlays the location where this datum is on. So does a *redo* operation. For a dimensional tolerance, because of its unique connection with its feature volume, it needs to be reexamined to check if its feature volume has been resized. Obviously, operation *redo*, *insert*, *delete*, *modify*, and *replace* are operations that may change the size of feature volumes.

Geometric tolerances are affected by all feature editing operations. The reason is that geometric tolerances are logically linked to the topological elements of a feature, such as a volume, a face, and an edge. Basically, modifications of a geometric tolerance could occur in following cases: first, an editing operation could remove a topological element from a product model. If this topological element has been used by another tolerance as its referenced object, that tolerance should be removed from both its own location and its referenced location. Second, a *modify* operation to a feature may invalidate some tolerances on this feature. For example, after a rotation is done on a form feature, a face which was originally perpendicular to a datum surface may no longer perpendicular to it. In this case, the perpendicularity tolerance needs to be removed

from this face. Finally, some feature faces may disappear during a feature editing operation. If so, the tolerances on these faces needed to be examined and even removed.

7 Object-Oriented Implementation

The proposed feature modeling system has been implemented using the geometric modeler ACIS and VERSANT object-oriented database system in C++. The system runs on a Sun Sparc-10 station under UNIX and X Window environments. In the following subsections, implementation issues of the system will be discussed such as object-oriented representation of feature model, main class definitions, and graphical user interface.

7.1 Object-Oriented Representation of Feature Model

In the view point of object-oriented analysis, a system can described with objects and messages. Objects define the states and behaviors of system components and messages define the interaction among objects. Every object may contain a set of attributes which define its states and/or a set of methods which are able to change its states. The methods of an object are invoked by a message passing from the outside or inside. Messages cause objects to communicate with each other and therefore change the system's behavior. Objects are described by classes.

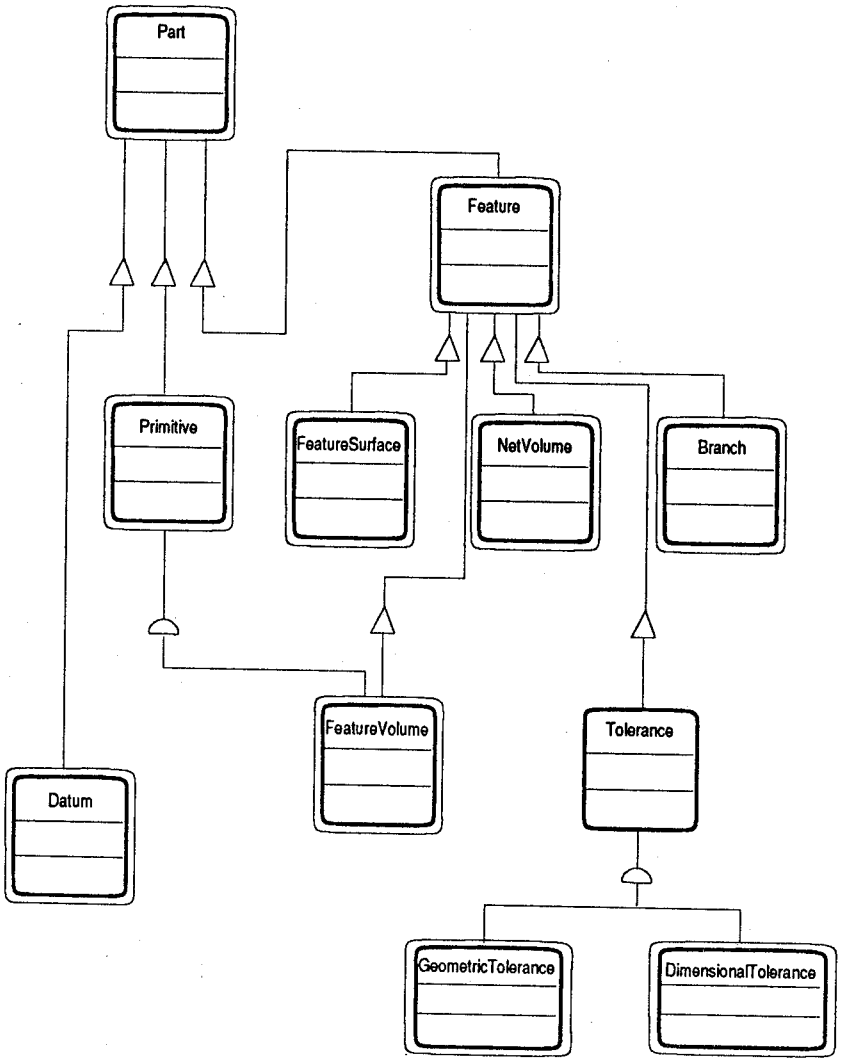


Figure 8: System Classes Hierarchy

With this concept, the design of the proposed feature modeling system is initiated by defining its objects and messages. The core class is, of cause, "feature". As mentioned in Chapter 4, the main feature attributes include 1) feature surface; 2) feature volume; 3) net volume; and 4) creating solid. Beside these basic attributes, each feature should also contain attribute branches which link the feature to its adjacent features and attribute tolerances which associate manufacturing specifications with this feature. Additionally, a part object is required to capture administrate data to the product and datum system above the feature level. These objects can be abstracted into nine classes, each of them capturing all objects that have the same behavior, using object-oriented analysis techniques. These nine classes are: **Part**, **Feature**, **FeatureSurface**, **FeatureVolume**, **NetVolume**, **Primitive**, **Branch**, **Tolerance** and **Datum**. The hierarchy

of these classes is shown in Figure 9. Figure 10 shows their definition. In Figure 9, each class is represented by a box. The class name is at the top cell of the box. The middle cell contains the attributes while methods are in the bottom cell. The relationships between classes are managed by edges with either a half-circle or a triangle. The half-circle edge represents is-a relationship and triangle edge captures has-a relationship.

As shown in Figure 9, a **Part** object has one or more **Feature** objects which consist of the feature graph. It also has a list of **Primitive** objects which are the primitive solids used to create this part model. Also included in this **Part** object is a list of **Datum** objects which define the datum system of this part. The relationships are shown in by three triangle edges from the class **Part** to the classes **Feature**, **Primitive**, and **Datum**, respectively. A **Feature** object has a **FeatureSurface** object, a **FeatureVolume** object, and a **NetVolume** object. These three components define the geometry of this feature. It also has two set of **Branch** objects which respectively link its parent adjacent features and child adjacent features. In additionally, it has a list of **Tolerance** objects each of which is either a geometric tolerance or a dimensional tolerance. The relationships between these five classes and class **Feature** are all maintained by has-a edges. The class **Tolerance** is defined as an abstract class. The objects of class **Tolerance** are either an object of its subclass **GeometricTolerance** or an object of subclass **DimensionalTolerance**. All product information can be captured concisely and comprehensively in this simple object-oriented representation scheme.

7.2 Class Definition

The attributes and methods of each above class are listed in Figure 10(a) through (i). In this subsection, the meaning of each of these attributes and the behavior of methods is explained.

As shown in Figure 10(a), the class **Part** includes the attributes part_id, part_name, part_version, base_part_shape, material_type, bounding_box, AISI_SAE_code, designer, design_date, feature_root, primitives, and datum. Among these attributes, the base_part_shape tells the shape of stock, the bounding box specifies the minimum dimensions of the finished product, and AISI_SAE_code is a standard material code. The feature_root is pointer to the root of the feature graph of defined part, an instance of class **Feature**. The datum is a pointer to a list of datums which are instances of class **Datum**. Finally, primitives is the pointer to a list of primitive solids that have been used in the creation of this part. These primitive solids are instance of class **Primitive**.

Main methods associated with the class **Part**, except inquiring methods, are *create_feature()* and *link_feature()*. The method *create_feature()* creates the geometry of a new feature by applying a Boolean operation between a primitive solid and this part object. It then calls methods in class the **Feature** to generate feature surface, feature volume, and net volume, while the function

link_feature() adds this newly-created feature to the feature graph by according to their face adjacent relationships with other existing features. The feature graph is updated by adding this new feature as the child of all its adjacent existing features through the instances of class **Branch**. The *link_feature()* function checks if all these adjacent feature are still actively linked to their original neighbors after this new feature is added. Meanwhile, it also examines the feature volume of the adjacent features and adjusts the dimensions of these feature volumes. Finally it also checks the feature type of all the adjacent features and renames them if necessary.

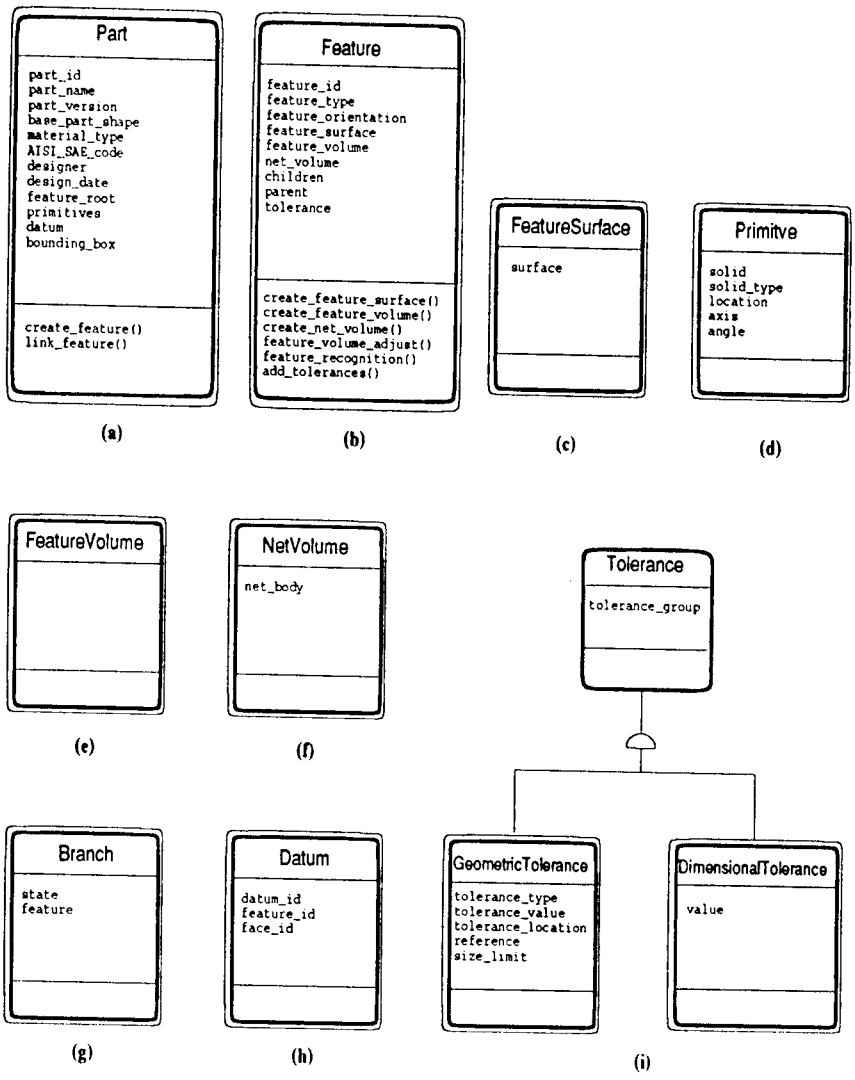


Figure 9: Classes Definitions

The class **Feature** shown in Figure 10(b) defines all nodes of the feature graph. It capture the attributes feature_id, feature_type, and feature_orientation. Other attributes include corresponding feature_surface, feature_volume, net_volume, creating_solid, children, parent, and tolerances. Feature_id is the unique identification of this feature in the part. It is the same as the instance ID mentioned in Chapter 4. Feature_orientation is the major direction of this form feature. It is the normal of face 0 on the bounding shell of its creating solid as shown in Figure 3. The feature_surface, feature_volume, net_volume, and creating_solid are pointers to the object of **FeatureSurface**, **FeatureVolume**, **NetVolume**, and **Primitive**, respectively. Parent and children are two lists of pointers pointing to the objects of **Branch**, which capture its adjacent features that were created earlier or later, respectively. The attribute tolerances is a list of pointers

pointing to objects of either the objects of **GeometricalTolerance** or **DimensionalTolerance**. Main methods defined in this class include *create_feature_surface()*, *create_feature_volume()*, *feature_volume_adjust()*, and *feature_recognition()*. The first three methods are invoked each time a new form feature is created. These methods create a feature volume, a feature surface, and a net volume according to the newly-created geometry. The methods *feature_volume_adjust()* and *feature_recognition()* function may be called frequently each time this form feature has changed its geometry. The former one makes sure that the feature volume keeps correct dimension and the later one ensures that this feature is correctly named. Through the method *add_tolerance()*, a dimensional tolerance or a geometric tolerance can be attached to the geometry of this feature at all levels. During the addition of tolerances, the tolerance validity checking discussed in Chapter 7 will be executed simultaneously.

Feature components are captured by the classes **FeatureSurface**, **NetVolume**, **Primitive**, and **FeatureVolume**. These classes are shown in Figure 10(c), 10(d), 10(e), and 10(f). The class **FeatureSurface** contains a set of face ids which identify this feature faces on the product. The class **NetVolume** includes the attribute net_body, which is a pointer to the B-rep which models the net volume of this feature. The B-rep is actually represented by an ACIS BODY. The class **Primitive** maintains all attributes required to represent a creating solid. These attributes include solid, solid_type, location, axis, and angle. The attribute solid is one of the primitive solids defined in Chapter 4. Its dimension is represented depending on its solid_type. For example, the dimension of a block is represented by its width, height, and length, while a cylinder is described by its radius and height. Attribute location defines translation parameters. The axis defines the axis along which the creating solid will be rotated by the magnitude given by the attribute angle. The class **FeatureVolume**, as shown in Figure 10(f), is defined subclass of the class **Primtive**, so it will inherit all attributes from the class **Primitive**. The reason why a new class is defined is that the attributes in the class **Primitive** will be used in a different way different from that in the the class **FeatureVolume**. In the class **FeatureVolume**, the attribute location will be used to capture the center point of the feature volume. The attribute axis is used to store the orientation of the feature volume, while the attribute angle is not used.

The class **Feature** shown in Figure 10(b) defines all nodes of a feature graph. It captures attributes of feature_id, feature_type, and feature_orientation. Other attributes include corresponding feature_surface, feature_volume, net_volume, creating_solid, children, parent, and Figure 10(g) shows the definition of class **Branch**. In this class, two attributes are defined as feature and state. The attribute feature is a pointer pointing to a **Feature** object. While attribute state denotes an active connection or inactive connection to this feature. In a feature graph, the active branches capture feature adjacency relationships between features, while inactive branches, together with active branches, keep track of feature modeling history. The class **Datum** is il-

lustrated in Figure 10(h). Attributes associated with this class are datum_id, feature_id, and face_id. Datum_id is a denoted by a single capital letter 'A', 'B', ... Datum location is defined by feature_id and face_id. At present time, both datum and geometric tolerance locations are limited to a volume or a face.

The last class is **Tolerance**. As shown in Figure 10(i), it has two subclasses: **GeometricTolerance** and **DimensionalTolerance**. The class **Tolerance** is defined as an abstract class. The objects of all its subclasses can be uniformly called instances of class **Tolerance**. The attribute defined in class **Tolerance** is tolerance_group, which distinguishes an object of subclass **GeometricTolerance** from an object of **DimensionalTolerance**. The attributes defined in the class **GeometricTolerance** are tolerance_type, tolerance_value, tolerance_location, reference, and size_limit. Among these attributes, tolerance_location is either a volume or a face, and reference is a list of pointers pointing to **Datum** objects or other locations (volume or face). The attribute in the class **DimensionalTolerance** is a value kept in a four dimension array. Based on the primitive solids defined in Chapter 4, only four parameters at most are needed to define a primitive solid.

7.3 Graphical User Interface

A graphical user interface (GUI) has been developed to provide a friendly interface to our proposed feature modeling system. As shown in Figure 11, this GUI is written based on a small set of GUI utilities which are developed with X11 library and C++. The GUI utilities contain various buttons, menus, message boxes, dialog boxes, and a view port. Buttons consist of *push button*, *check button*, and *radio button*. A *push button* sends an ON message to controller when it is clicked and released. The *check button* and *radio button* act the same as a *push button* but are usually used in conjunction. For a group of *check buttons*, only one of them can be set up at a time. In the case of *radio buttons*, several buttons can be set up at the same time. The menu may either be a *pulldown menu*, a *cascade menu*, a *popup menu*, or a *menu bar*. Both the *pulldown menu* and *cascade menu* pop up one or more submenus under current menu or to the right side of current menu, respectively. The *popup menu* pops up a dialog box, while the *menu bar* arranges several menus in a row or in a column. Both buttons and menus can be customized with either text, icons or both. The message box pops up a window with messages. It disappears after a confirm command is issued. The dialog box could be a confirm box or a text dialog box. The confirm box contains a short message with "OK" and "Cancel" buttons. This message is confirmed if the "OK" button is clicked. Otherwise "Cancel" button should be selected. The text dialog box is a pop up window with one or more string editors, an "OK" button, and an "Cancel" button. The string editor(s) pass user input to system after the "OK"

button is clicked. Finally, the view port window is a canvas on which various drawings may be displayed. The visible portion of a view port is surrounded by a frame. This frame can be moved on the canvas so that all canvas can be viewed. The drawing on the canvas can be scaled by enlarging it or reducing it. It can also be viewed from different perspective or orthogonal viewpoints. All these controls are shown in the Figure 11 at the lower portion in the column of the interface. The circles in that portion stand for the hemispheres at the up half-space of the plane XOY and the low half-space, respectively. The icon buttons in the higher portion of the column are the primitive solids defined in the system. When any of these buttons is clicked, an instance of the primitive solid is created and ready to be used to model a product.

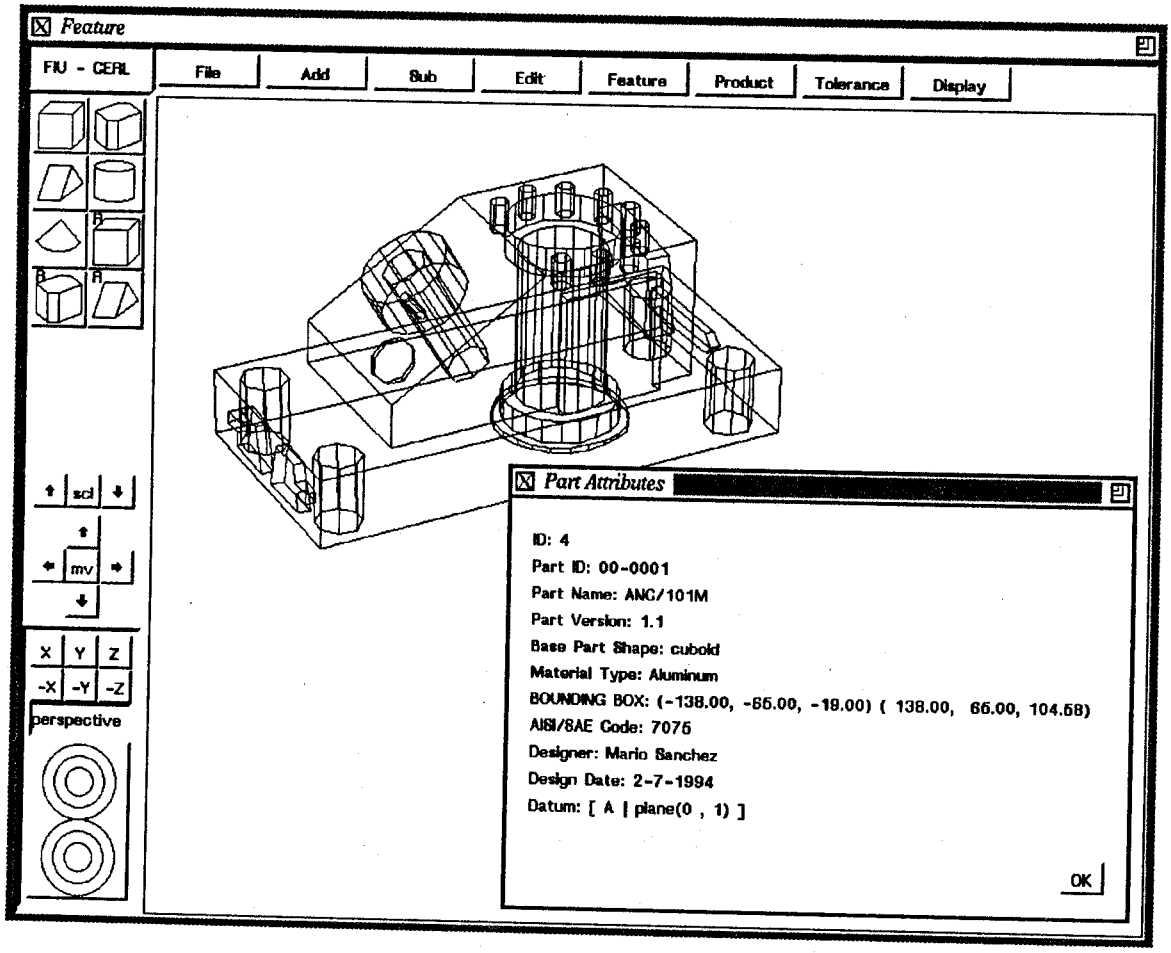


Figure 10: Graphical User Interface of Feature Modeling System

Figure 12 illustrates all menus in the row menu bar shown in Figure 11. There are eight main menus defined. They are *File*, *Add*, *Sub*, *Edit*, *Feature*, *Product*, *Tolerance*, and *Display*. *File* has five submenus: *New*, *Load*, *Delete*, *Save*, and *Quit*. They are respectively used to create a new product from a CSG data file, load a product from the shared database, delete current

product from shared database, persistently store the current product into the shared database, and quit from the feature modeling system. The menus *add* and *sub* carry out a Boolean addition and a Boolean subtraction operation between an instance of primitive solid and current product. Under menu *Edit*, submenus *Undo*, *Redo*, *Insert*, *Delete*, *Replace*, and *Modify* are defined. These menus invoke a feature editing operation on an active form feature which is clicked from the current product.

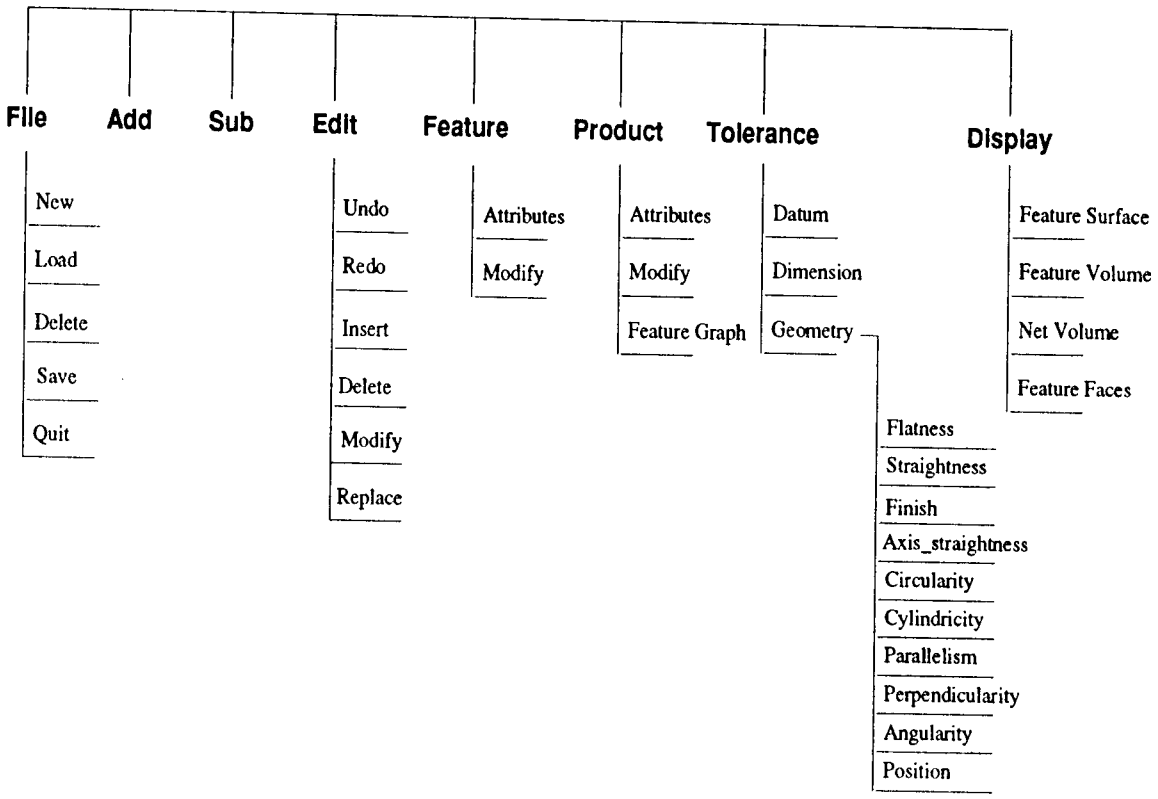


Figure 11: System Menu Layout

The menu *Feature* has two attributes which are *Attributes* and *Modify*. The former one displays the attributes of a currently active feature and while the latter allows the designer to modify feature attributes. An example of feature attributes for the feature **step** is shown in the window named "Feature Information" in Figure 13. The menu *Product* has another submenu called *Feature Graph*, plus submenus *Attributes* and *Modify*, which are the same as the ones in *Feature* (an example of product attributes is shown in Figure 11). Clicking on these submenus will pop up a sub-window which shows the feature graph of the current product. The detailed information about individual features can be accessed by clicking feature nodes in the feature graph, as illustrated in Figure 13.

The menu *Tolerance* contains submenus *Datum*, *Dimension*, and *Geometry*. The first one

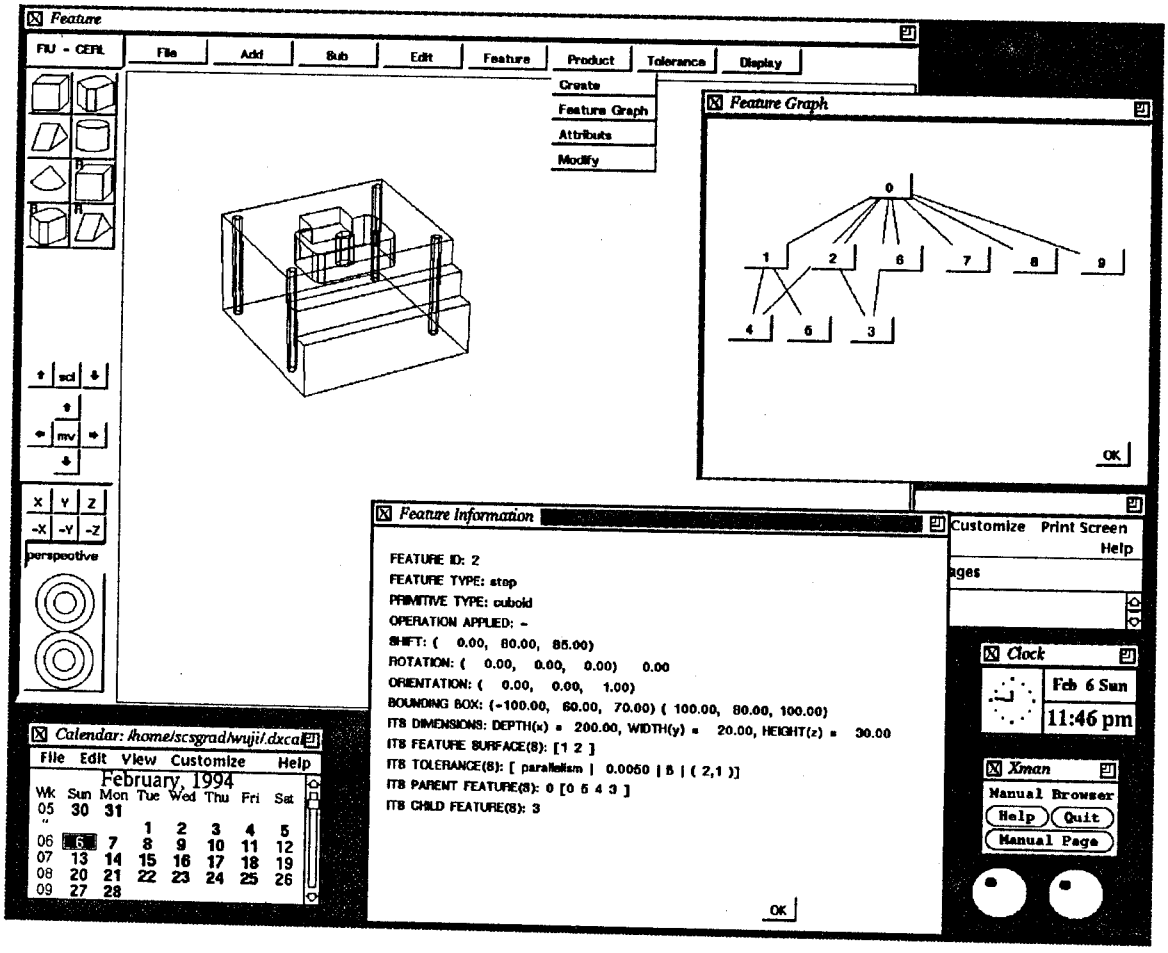


Figure 12: Feature Graph and Feature Information Shown on GUI

allows the designer to assign a datum to a selected topological element such as a face or a volume from the product shown on the screen. The second one specifies a dimensional tolerance to the currently active form feature. The third one is used to specify various geometric tolerances to current product. The available geometrical tolerances are listed as the submenus under this menu. These submenus are corresponded to the ten geometric tolerances defined in Chapter 7. Finally, under the menu *Display*, four submenus *Feature Surface*, *Feature Volume*, *Net Volume*, and *Feature Face* are defined. These menus are used to change the display mode when a form feature is selected on current product. The first three will display the feature surface, the feature volume, and the net volume of a feature respectively when one of its faces is clicked. If the display mode is set to *Feature Faces*, only the clicked face will be highlighted.

8 Conclusion

A new hybrid CSG/B-rep feature modeling system which supports feature recognition and feature editing has been proposed. In addition to volume and surface information of a form feature, the proposed system captures the shape information of each form feature at modeling stages and tightly associates this information with its topological elements. Meanwhile the feature shape information is organized into a feature node and added into the feature graph. The proposed feature modeling system also capture manufacturing data such as dimensional tolerances, geometric tolerances, and datums by attaching them to the product geometry at all levels through the feature graph.

With the feature shape information, all features in the product can be unambiguously recognized by using two formats of simple feature rules. The advantage of this approach is that a new feature rule can be automatically generated during a recognizing process. Therefore, the feature extraction capability can increase gradually during the modelings. To support feature editing operations, the proposed system also keep the track of feature modeling history by maintaining both active and inactive adjacency relationships between form features. By applying feature modeling history, six feature editing operations have been defined. The impacts of these feature editing on dimensional tolerances, geometric tolerances, and datums are investigated and solutions are given.

Unlike existing feature modeling systems in which feature modeling is separated from geometric modeling, the proposed feature modeling system uniformly combines both modelings in an integrated CAD/CAM environment. In this system, pre-defined features are no longer the bottle neck as that in existing feature-oriented systems. They can not only be automatically added during the modeling but also be edited in both geometry and semantics. The proposed feature modeling system, therefore, is very powerful and capable for CAD/CAM integration.

References

[1] K. Case and J. Gao. Feature Technology: An Overview. *International Journal of Computer Integrated Manufacturing*, 6(1&2):2–12, 1993.

[2] T. Chang. *Expert Process Planning for Manufacturing.* Addison-Wesley Publishing Company, Inc., 1990.

[3] J. Corney and D. E. R. Clark. Efficient Feature-Based Feature Recognition. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of the Second ACM/IEEE Symposium on Solid Modeling and Applications*, pages 313–322, Montreal, Canada, 1993. The ACM Press.

[4] L. de Floriani and B. Falcidieno. A Hierarchical Boundary Model for Solid Object Representation. *ACM Transactions on Computer Graphics*, 7(1):42–60, January 1988.

[5] I. A. Donaldson and J. R. Corney. Rule-Based Feature Recognition for 2.5D Machined Components. *International Journal of Computer Integrated Manufacturing*, 6(1&2):51–64, 1993.

[6] A. D. Fleming. A Representation for Geometrically Toleranced Parts. In J. Woodwark, editor, *Geometric Reasoning*, pages 141–163. Oxford Science Publications, 1989.

[7] A. J. P. Gomes and J. C. G. Teixeira. Form Feature Modeling in a Hybrid CSG/B-rep Scheme. *Computer and Graphics*, 15(2):217–229, 1991.

[8] M. R. Henderson. *Extraction of Feature Information From three-Dimensional CAD Data.* PhD thesis, Purdue University, 1984.

[9] J. Hwang and M. Henderson. Applying the Perceptron to 3D Feature Recognition. In *Proceedings of NSF Design and Manufacturing Systems Conference*, pages 703–707, Atlanta, GA, January 1992.

[10] S. Joshi, N. N. Vissa, and T. Chang. Expert Processing System with Solid Model Interface. *International Journal of Production Research*, 26(5):863–885, May 1988.

[11] L. K. Kyprianou. *Shape Classification in Computer Aided Design.* PhD thesis, Cambridge University, Engineering Department, 1980.

[12] Y. Lee and K. Fu. Machine Understanding of CSG: Extraction and Unification of Manufacturing Features. *IEEE Transactions on Computer Graphics and Application*, 17(1), 1987.

[13] B. Liou. *Pseudo Boundary Model and Feature Interface for Design by Features.* PhD thesis, Mechanical & Aerospace Engineering Department, Arizona State University, 1988.

[14] M. J. Pratt. Synthesis of an Optimal Approach to Form Feature Modeling. In *ASME Computer and Engineering Conference*, pages 263–274, San Francisco, California, August 1988.

[15] M. J. Pratt. Solid Modeling — Survey and Current Research Issues. In D. F. Rogers and R. A. Earnshaw, editors, *Computer Graphics Techniques: Theory and Practice.* Springer-Verlag, New York, 1990.

[16] M. J. Pratt and P. R. Wilson. Requirements for Support of Form Features in a Solid Modeling System. Technical Report R-85-ASPP-01, CAM-I Inc., Arlington, Texas, USA, 1985.

[17] M. Ranta, M. Inui, F. Kimura, and M. Mantyla. Cut and Paste Based Modeling With Boundary Features. In J. Rossignac, J. Turner, and G. Allen, editors, *Proceedings of the Second ACM/IEEE Symposium on Solid Modeling and Applications*, pages 303–312, Montreal, Canada, 1993. The ACM Press.

[18] A. A. G. Requicha. Solid Modeling and its Applications: Progress in Tolerancing, Inspection, and Feature Recognition. In *Proceedings of NSF Design and Manufacturing Systems Conference*, pages 5–17, Tempe, Arizona, January 1990.

[19] A. A. G. Requicha and S. C. Chan. Representation of Geometric Features, Tolerances, and Attributes in Solid Modelers Based on Constructive Geometry. *IEEE Journal of Robotics and Automaton*, RA-2(3):156–166, 1986.

[20] J. R. Rossignac. Issues on Feature-Based Editing and Interrogation of Solid Models. *Computer and Graphics*, 14(2):149–172, 1990.

[21] U. Roy and C. R. Liu. Feature-Based Representational Scheme of a Solid Modeler for Providing Dimensioning and Tolerancing Information. *Robotics and Computer-Integrated Manufacturing*, 4(3/4):335–345, 1988.

[22] J. J. Shah and M. T. Rogers. Expert Form Feature Modeling Shell. *Computer Aided Design*, 20(9):515–524, November 1988.

[23] J. J. Shah, P. Sreevalsan, M. Rpgers, and A. Mathew. Current Status of Features Technology. Technical Report R-88-GM-04.1, CAM-I Inc., Arlington, Texas, USA, November 1988.

[24] N. Wang and T. M. Ozsoy. A Scheme to Represent Features, Dimensions, and Tolerances in Geometric Modeling. *Journal of Manufacturing Systems*, 10(3):233–240, 1991.

[25] C. S. Chen.and J. T. Wu. A hybrid CSG/Brep representation scheme for feature modelling. *CSG 94*,291-303, 1994

Appendix D: Producibility Evaluator

Appendix D: Producibility Evaluation

D1. Introduction

The producibility evaluation methodology developed in this study adopts the form feature concepts and object-oriented computer programming and database techniques. framework of this evaluation methodology is configured in Figure 1. It is structured to examine many aspects of producibility can be examined and to provide the product designer with feedback such that the advantages of concurrent engineering can be utilized during the design stage. This figure illustrates the relationship of the producibility evaluation framework to the overall concurrent engineering scheme. Note that a producibility evaluation can be conducted both before and after a process plan is generated. Before process plan generation, an evaluation of the product design has to rely solely on the product model and the manufacturing resources models. Cost, quality and schedule evaluations require more information to be accurate, and thus should be performed after process planning.

D1.1 Producibility Evaluation Framework

The proposed framework for producibility evaluation consists of two evaluations prior to process plan generation and three evaluations after process plan generation. An outline of the framework, as per Figure 1, is provided as follows:

I. Prior To Process Plan Generation.
   A. Check Technical Feasibility.
   B. Calculate Producibility Index.
II. After Process Plan Generation.
   A. Estimate Life-Cycle Cost.
   B. Calculate Process Capability Index.
   C. Assess Schedule Compatibility.

These five modules answer fundamental manufacturing questions typical of any given product: *(1) can it be made?; (2) how easily?; (3) how much will it cost?; 4. what is the expected quality level?; and (5) can it be made in time?*

The first phase of the evaluation serves as direct feedback to assist the designer by providing manufacturing insights, while the second phase provides accurate risk data of particular interest to upper management. Technical feasibility assessment must precede calculation of the producibility index in order to know whether the calculation is necessary. The technical feasibility check is made to determine if the product is manufacturable with the available facilities. This check is done by identifying a feasible stock material and by identifying a feasible machine and tool combination for fabricating each feature. If no machine or tool can be identified for any single feature, then the entire design is assessed to be infeasible and redesign of the part or new equipment is mandated. One may also consider a more comprehensive check against all known manufacturing resources, but this would require a much larger database. If a product

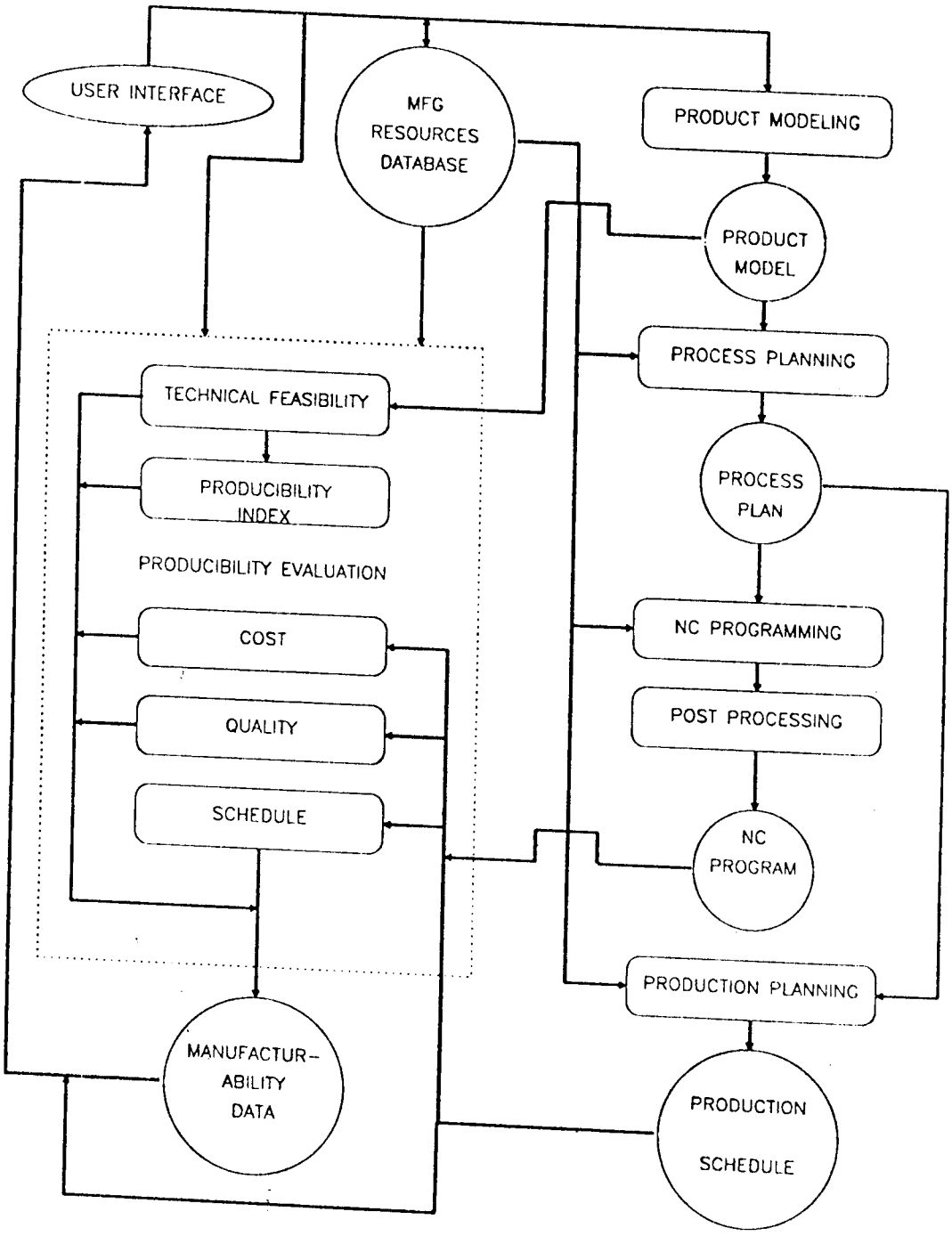


Figure 1: Computer Integrated Concurrent Engineering.

is deemed feasible, then one needs to rate its producibility. A producibility index based on difficulty criteria is used for rating the product. Difficulty is inferred from the states of a selected set of part design parameters. A detailed development of the producibility index for machined components will be provided in section D3.

Cost and quality metrics are derived from the process plan, and schedule compatibility is derived from a production schedule. Regardless of the product design, the cost is derived from a process plan. Unit costs are used for equipment and tooling, processing times and material cost. An activity based cost model expanded to the life-cycle level is considered the best approach. In order to calculate a process capability index, C_{pk}, the process capability of each piece of manufacturing equipment must be captured. Then, with the aide of a process plan for identifying all processing equipment, one may retrieve all necessary information for calculating a C_{pk} index for the product. Since the quality of a product can be affected by the performance of a single piece of equipment, one should identify the least capable machine and use it for calculating the C_{pk}. The least capable machine would be identified as the machine with the smallest margin for meeting tolerance requirements assigned to it. For example, consider a drill that is assigned to make a hole that has no tolerance or surface finish requirements, as opposed to a milling machine that is assigned to make a pocket with tolerances specified near the limits of the machine's capability. In this case, the milling machine has the smaller margin, even though the milling machine is generally more capable.

Assuming that all equipment and tooling availability is known, that all existing jobs are known according to process plan, due date, and production volume, then the through-time is computed for each job. This through-time can then be used with start-time and end-time requirements for the part for schedule compatibility evaluation.

D1.2 Selection of a Framework Subset for Development

Technical feasibility assessment and calculation of the producibility index represent a sort of self-contained aspect of the overall producibility evaluation framework. Development of the entire framework is beyond the scope of this thesis, therefore attention will focus on the phase prior to process plan generation, as outlined.

A detailed description of the proposed method for producibility evaluation prior to process plan generation is presented next. The description is supported by various figures, tables and equations where necessary. The implementation results are supported by example evaluations of test parts. Technical feasibility is described first, followed by a description of the producibility index computation algorithm.

D2. Technical Feasibility Assessment

Within the context of this thesis, technical feasibility assessment will be applied to features that are oriented perpendicular to their contact face. A feature under this condition is called a 1D-orientation feature. A 2D-orientation feature would meet its contact face at an angle other than 90°, and would lie in a plane normal to the contact face. A 3D-orientation feature is the default case if not 1D or 2D. Figure 2 illustrates these three orientations. Some important feasibility issues will be addressed next.

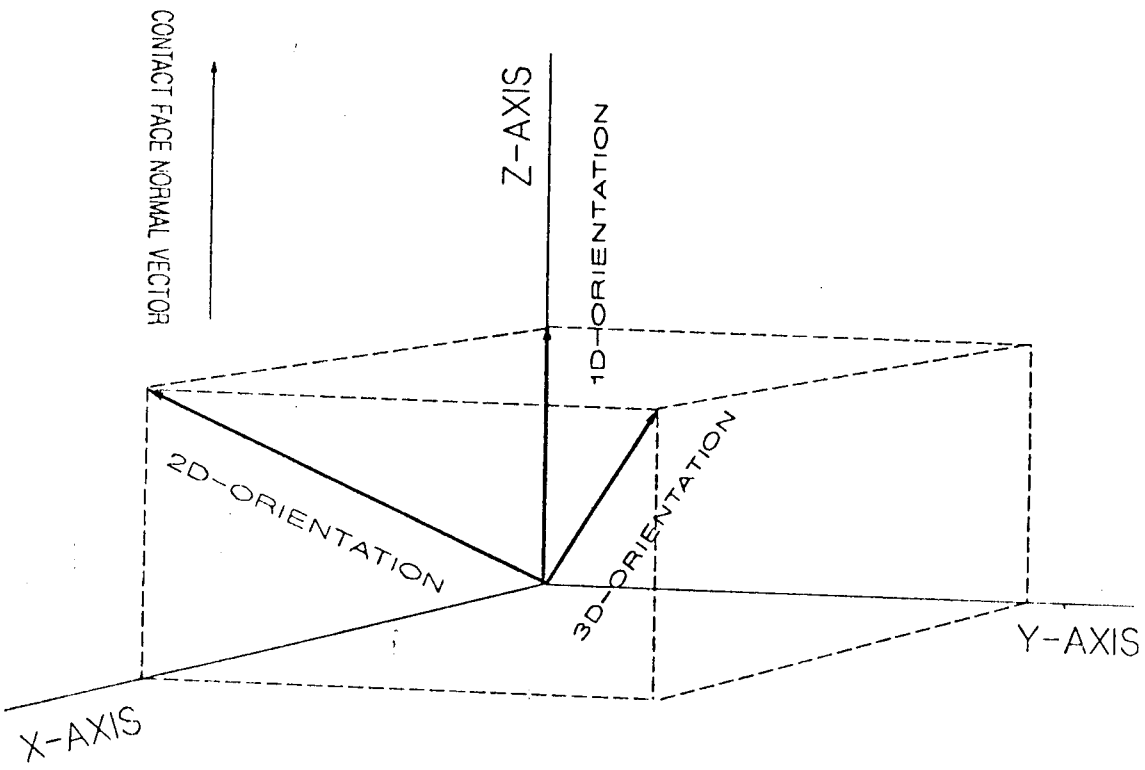


Figure 2: Classifications of Form Feature Orientations.

There are three primary concerns with regard to machine tool capacity; workspace capacity, power capacity, and load capacity. Workspace capacity is concerned with fitting the part on a machine. If a part is too large to fit on any machine in the plant, then either a new larger machine must be purchased or the part must be modified. The size of the part alone may not provide enough information, because the fixturing elements can effectively increases part size, and the tool assembly effectively decreases the workspace size.

Gopalakrishnam, et al. [1] described a method for calculating the material removal rate(MRR) via a regression equation that applies to face milling:

$$MRR = e^{-[0.13224 - 0.000013BHN^2]}$$

where BHN is the Brinell hardness number. Empirical formulas that relate material removal rate to horsepower can be used to estimate the power requirements of the candidate machine. The material removal rate is considered the optimum or near the optimum rate, which implies other rates may be feasible. The size of a milling cutter body and its number of inserts influences the power requirements and material removal rate. Since the minimum possible required horsepower is likely to define the feasible power requirement, this type of check could become cumbersome due the number of parameters involved, and is not considered for this development. The load capacity of a machine is the amount of weight that the machine's table can accommodate without affecting the machine's ability to operate as designed. Workspace capacity and table weight capacity are considered in this development.

A tool assembly is basically a cutter (with or without inserts) and any additional elements that are necessary for linking the cutter to a given machine in such a manner that the composite assembly is suitable for its intended application. Additional assembly elements include holder, adapter, and extension. The holder is the interface between the cutter and machine, if the machine has no device such as a chuck. If a cutter is not directly compatible with a holder then an adapter is required. If a cutter is not long enough then an extension is needed. Some cutters are solid body cutters such as a common drill bit, while other cutters use inserts that effect the chipping action. Some tool assemblies are not cutters, but grinders. Form tolerances and surface finish, as well as machining parameters are affected by tool selection. Tool selection is complicated by conflicting demands for efficient material removal and minimal tool changes, according to Bala and Chang[2]. This occurs when one is faced with irregular features, especially pockets, that allow different size cutters at different locations within their boundaries. Material removal efficiency suggests that an assortment of cutters should be selected in order to reduce the machining time and the number of passes. On the other hand one wishes to minimize tool changes, because tool changing is not a value adding activity, which complicates the issue of which tool(s) to choose. Cutter selection for making a pocket is accomplished by Bala and Chang by following the outer profile with an imaginary cutter such that the cutter center is offset by the cutter radius. Then a resultant feasible zone is created from which the cutter can remove all of the material. The largest cutter is chosen in this manner, and then a finish cutter that can follow the

entire outer perimeter is used for completion of the pocket. This method is also used for pockets with islands. Maximizing the cutter size reduces machining time, at the expense of only one tool change for many cases. By using this method one could simultaneously select the best tool combination and obtain a measure of the producibility of the feature in terms of the number of tool changes required and the machining time. A feasible tool would be a tool that is capable of producing the finished feature. This development considers the selection of a single feasible tool for each form feature, for the sake of simplicity, and does not deal with identifying an optimal tool set, which would be beyond the requirements of a feasibility assessment.

Tolerance is one of the most critical machining factors and is strongly considered in this development. When one tolerances an entity such as a machined part one is specifying the amount of acceptable deviation from nominal for that part. Tolerances are subdivided into tolerance categories and classes, as described next per Ranyak and Fridshal [3]. *Category* groups tolerances according to commonality and *class* defines the individual tolerance types in a category. For example, *form* is a category that *circularity* belongs to. A *tolerance data field* is used to document each tolerance entity and includes information such as system name, user name, class, and attached data. For example, consider a mechanical tie rod with a circular mounting hole at one end, the tolerance data field for radius could be: Tie Rod Mounting Hole, Radius(+.003 -.001)in. There may be variations on this format. A *data reference frame* is a type of feature composed of tolerance features that define a particular coordinate frame. A tolerance class such as circular runout would require a data reference frame. If the tie rod discussed above were toleranced for circular runout, then the data reference frame would be the center line along the length of the rod. A *tolerance qualifier* specifies what aspect of a feature is to be toleranced For example, for the tolerance on the primary cross section of a feature one would use the defining cross section, such as the area normal to the major axis of the feature. A feature is resolved into the defining primitive that best represents the tolerance to be evaluated. An example of this resolution would be the use of the geometric center point of a feature for tolerancing position. Here the feature is *resolved* to a defining point and that point is *evaluated* with respect to the deviation of its position from the nominal value. Tolerances and surface finish are assigned to individual form features, rather than being treated as global parameters, which allows one to capture more precise information regarding these parameters. Attaching tolerances and surface finish to features is essential to a feature-based method, and is this approach is adopted for this development.

Fixturing is an important feasibility issue. If a workpiece can not be fixtured then the part can not be machined and the part is not feasible. Most parts can be fixtured, or at least clamped, but extremely brittle parts, soft or flexible parts, and parts with unusual geometries may be too difficult to constrain. For the purpose of this development, fixturing is assumed to be either by means of a vise or some other simple method. This assumption seems reasonable, because the parts under consideration are prismatic and cuboid (except for sheets, which are clamped).

The feasibility assessment is made according to Figure 3, and is broken up into smaller detailed segments. As illustrated in Figure 3, feasibility is checked at the part

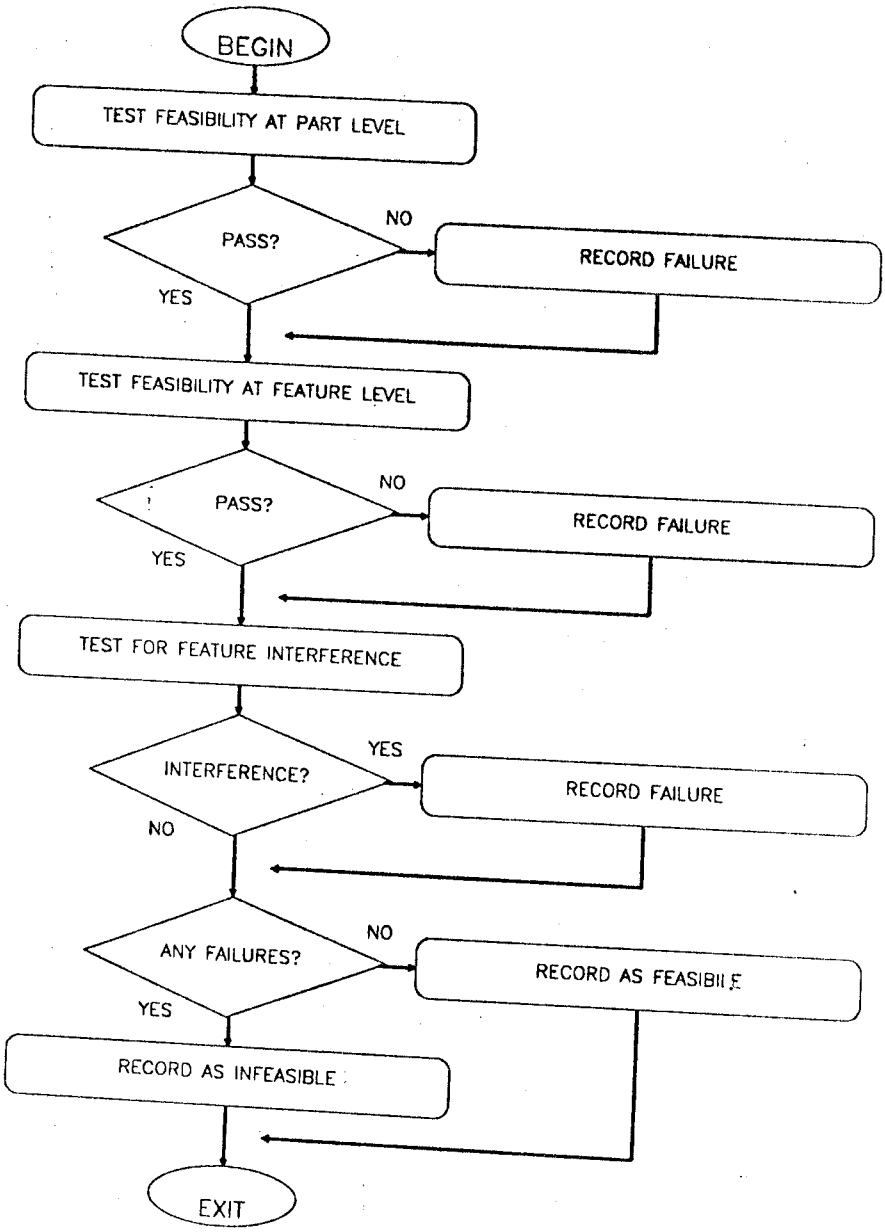


Figure 3: Technical Feasibility Assessment.

level, at the feature level, and with regard to interference between form features. This allows for capture of information regarding the nature of the feasibility of a part design, because failures are flagged at each stage of the assessment. If the part design is found to be infeasible at any one of the three stages, then it is assessed as infeasible. The three stages of the feasibility check are described in detail later in the text, beginning with the part level assessment, which is discussed next.

D2.1 Feasibility Assessment at the Part Level

Technical feasibility is first checked at the part level. Figure 4 illustrates the general approach. The link to the product and manufacturing resources data is made in the part level check because it is the first in succession. The first goal of the part level assessment is to locate a suitable piece of stock from which to fabricate the component. If no stock is found the part design is flagged infeasible and the part level assessment is terminated, else the stock's ID is recorded and the assessment continues. The second goal of the part level assessment is to identify a feasible machine & tool assembly set for processing the stock. If a feasible machine/tool combination is located their IDs are recorded and the procedure terminates, else the part design is flagged infeasible and then the procedure terminates. To summarize, the part level check provides the initial link to the product and manufacturing resources data, identifies a stock or indicates that no stock could be found, and identifies a machine/tool combination or indicates that none could be found.

Care must be taken when selecting stock material for a feasibility assessment. To be selected the candidate stock must be of the same material type, have the same AISI code, and have the minimum cross-sectional area of all feasible stock pieces. The stock cross-section is minimized so that a definitive test can be made with regard to machine work envelope capacity. If the smallest stock (cut to length) can not fit on a machine, then the machine is not feasible. A search of all stock classes is made and the minimized stock emerges as a result of constantly replacing the current candidate with stock of smaller cross-section. The bounding box of the part is used as a basis for comparing the dimensions of a piece of stock with those of the part. The bounding box is the smallest rectangular volume that can completely enclose the part. The dimensions of the bounding box are assigned to length, width, and thickness such that length is the largest dimension, width is the next largest and thickness is the minimum dimension. Once the part dimensions are set up for comparison with the stock, individual instances of stock are selected for screening. The procedure for selecting a piece of stock begins with rectangular classes and progresses through all classes of stock, currently rectangular, cylindrical, octagonal and hexagonal. Every qualified instance of every class of stock is checked so that the smallest piece is selected. Instances within a shape class qualify for checking only if their AISI material code is the same as that specified in the product model. First the stock is compared to the part enveloped to determine if it is large enough, then the cross-sectional area is compared with the current minimum to determine if the newly selected piece should replace the current candidate. The procedure begins with a high value for the initial minimum cross-sectional area as a starting point for the minimization routine. Once all stock has been evaluated the

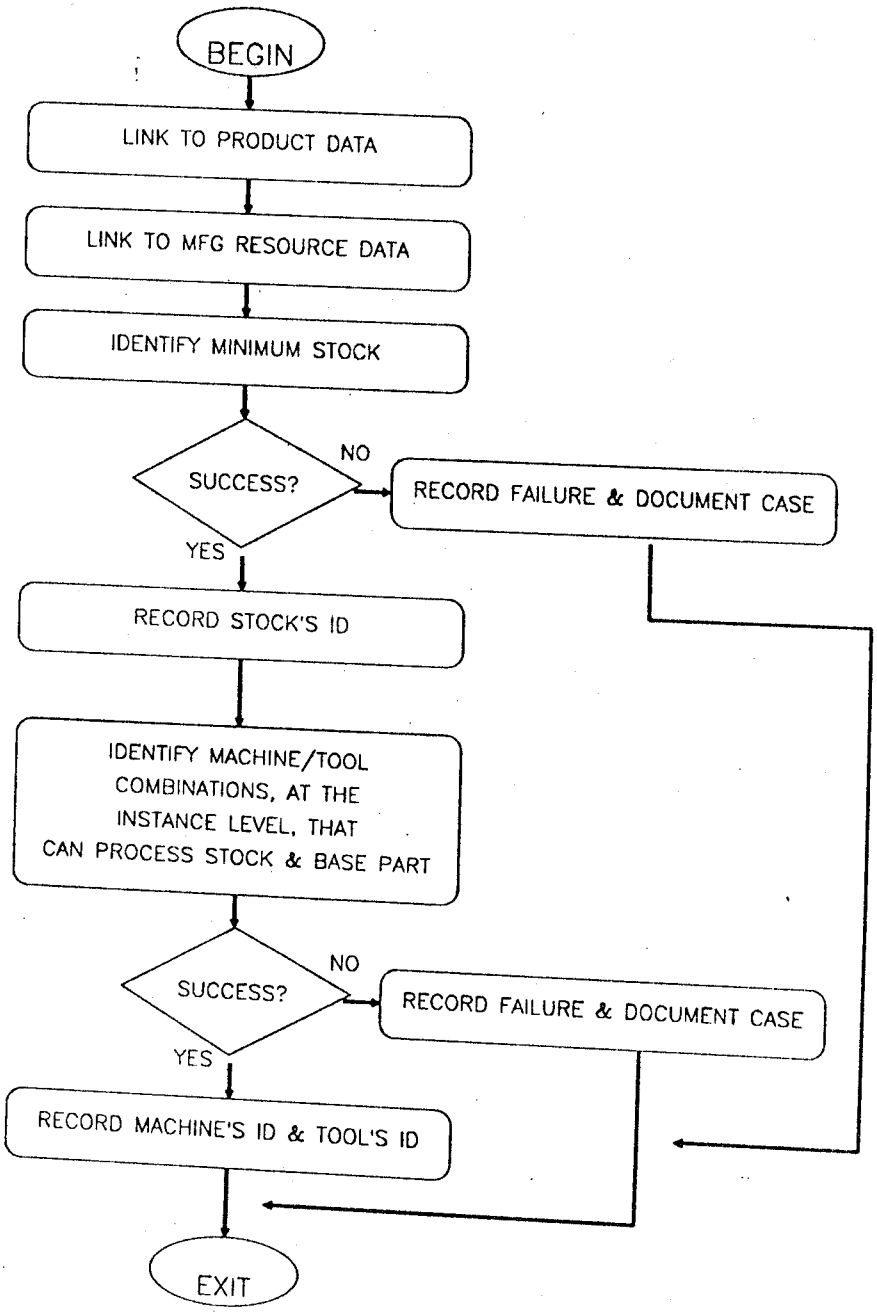


Figure 4: Technical Feasibility at the Part Level.

procedure is finished and the ID of the minimum stock is known. Figure 5 outlines the procedure for selecting the minimum stock, as just described.

In order to determine whether or not the stock is large enough, a simple approach was taken. Refer to Figure 6, which details the geometries of a few selected stock shapes. The first check is made between the stock length and the part envelope length. If the stock is at least as long as the part envelope it is acceptable. The next check is made between the part envelope's cross-sectional dimensions and those of the stock. For the case of cylindrical stock, the stock is acceptable if the part envelope diagonal is less than or equal to the diameter of the stock. For the case of rectangular stock, the stock is acceptable if the part envelope's thickness and width are less than or equal to the thickness and width of the stock, respectively. For the case of hexagonal and octagonal stock, the part envelope's thickness must be less than or equal to the flat distance of the stock, and the part envelope's width must be less than or equal to that of the stock. Examination of the hexagonal and octagonal shapes reveals that they may be capable of containing part envelope cross-sections with dimensions that do not meet the stated criteria for acceptability. Total accuracy in selection would require the use of a routine for checking that the part envelope does not intersect with the boundaries of the stock. This routine was not employed in order to maintain simplicity.

The next activity in the part level check involves the selection of a feasible machine/tool combination for processing the stock, which is made according to Figure 7. A machine class is selected first by using the Feature-Machine-Tool (FMT) table, which allows for cross-referencing between machine, tool and feature classes. The base-part is converted to a feature class that corresponds to one that is represented in the FMT table. Typically, the base-part is considered a cuboid boss. This is the feature type that is used for the base-part when selecting machines and tools from the FMT table. Note that though the base-part is cuboid in this development, the methods described herein are applicable to other base-part geometries, such as prisms, cylinders, etc. Each instance of this class is tested until one passes all the tests, or until all instances are exhausted. If all instances are exhausted the FMT table is revisited and the previous steps repeated. If a machine instance is selected, then a tool class is selected from the FMT table and the machine instance is tested simultaneously with instances of the tool class. If all tool instances fail, then a new tool class is selected. If all instances of all tool classes fail with the machine instance, then a new instance is tested. If all machine instances of all machine classes specified by the FMT table fail, then the part fails. There are two groups of tests that are made. The first group of tests is concerned only with machine aspects of technical feasibility, while the second group considers parameters that involve either the tool alone or the tool in conjunction with the machine. These two groups of tests are described next.

Figure 8 illustrates the machine related tests, which are the part weight test, the work envelope test, and the position accuracy test. If the cut-to-length weight of the stock is greater than the table weight capacity of the machine, then the machine fails the part weight test. If the magnitude of the tightest tolerance specification is less than the position accuracy of the machine, then the machine fails the position accuracy test. If the minimum work envelope dimensions are less than the largest stock dimensions, then

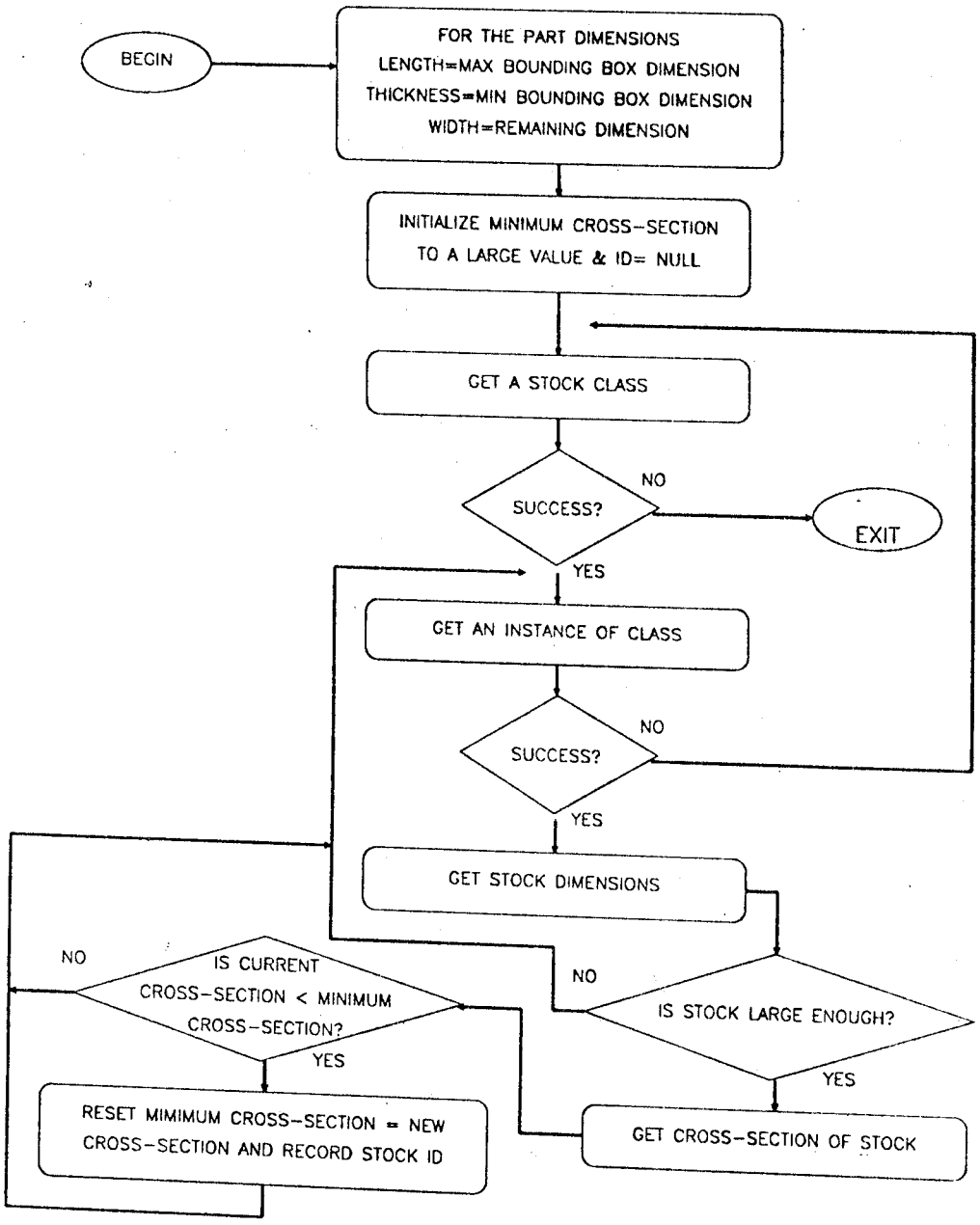


Figure 5: Select Minimum Stock.

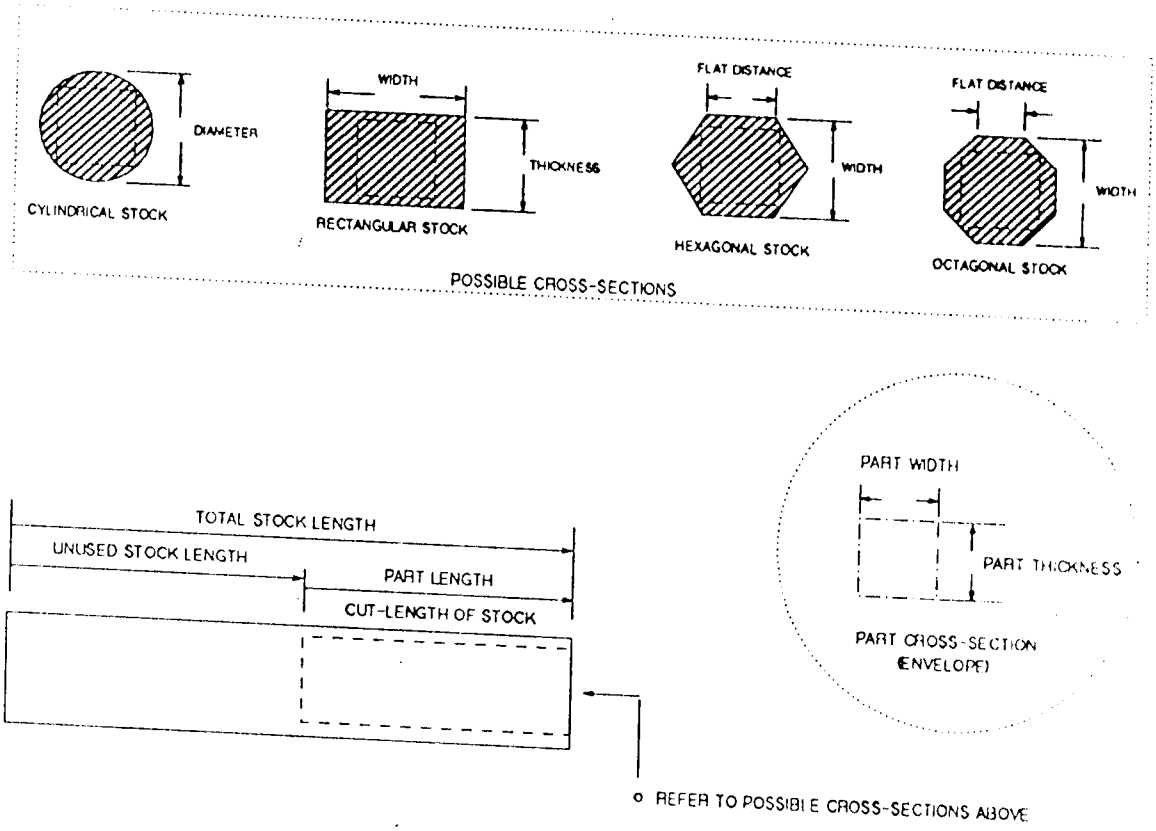


Figure 6: Illustration of Stock Selection.

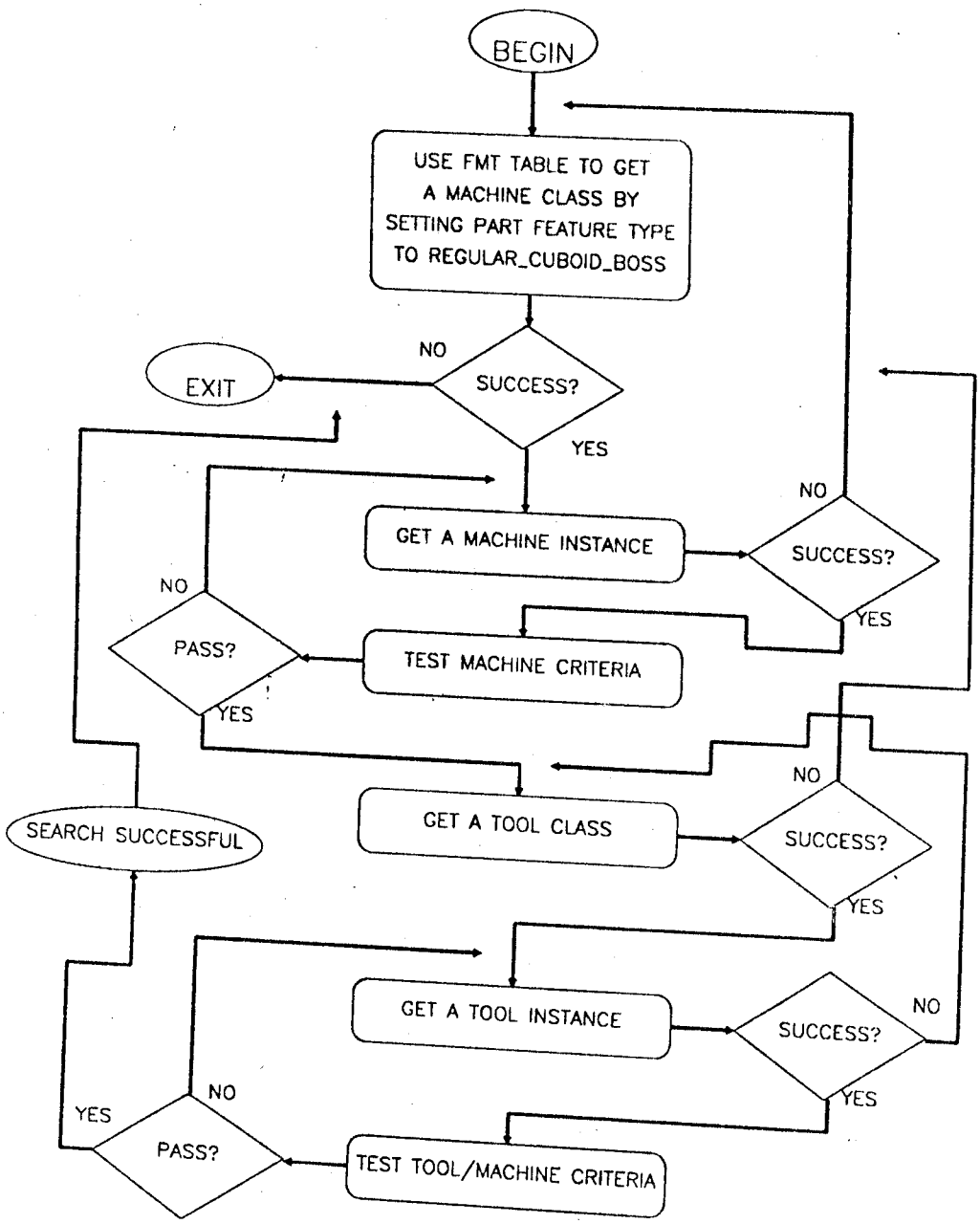


Figure 7: Identify a Feasible Machine and Tool for the Stock.

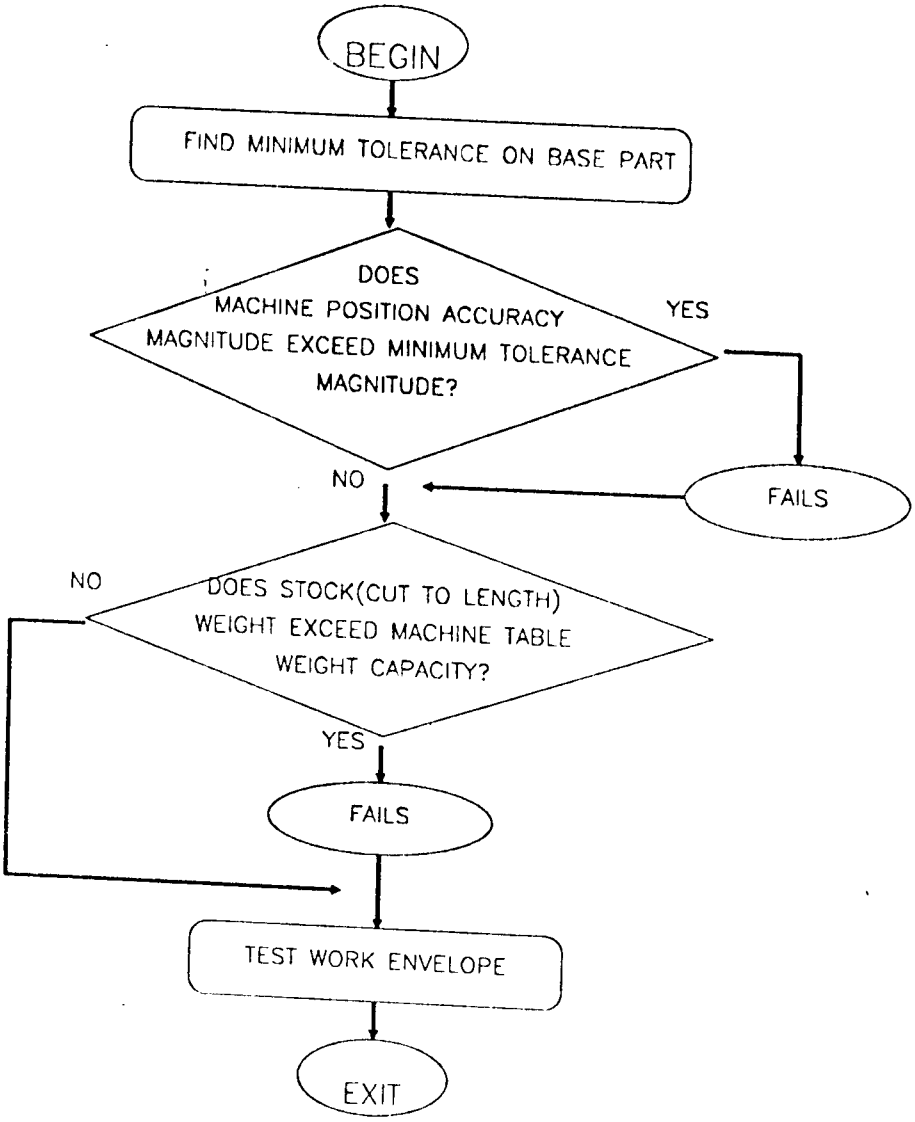


Figure 8: Test Machine Related Criteria at the Part Level.

the machine fails the work envelope test. The work envelope test will be discussed in more detail later in the text.

Figure 9 illustrates the tool & machine related tests, which are machine/tool compatibility test, material cut capability test, and surface finish capability test. If the tool holder type of the machine and tool do not match, then the tool fails the machine/tool compatibility test. If the magnitude of the minimum specified surface finish is less than the surface finish capability of the tool, then the tool fails the surface finish test. If the AISI code of the part material is not found in the tool's material cutting capability list, then the tool fails the material cutting capability test. Surface finish and tolerance specifications are taken from the base part.

D2.2 Feasibility Assessment at the Feature Level

The feature level feasibility assessment is similar to but more complex than the part level assessment. The general procedure for the feature level check is outlined in Figure 10. The procedure begins with the selection of the first feature after the base-part. A machine and tool class combination is then selected from the FMT table based on the feature type. Then the machines and tools are tested at the instance level. The machine instance alone is tested first, then tool and machine are tested. If all machine instances and tool instances of all classes specified by the FMT table fail, then the part is not feasible with respect to the current feature in question. The procedure is followed for each form feature in the product model. This provides the user with the capability of focusing on individual form features during design, which may allow a part design to be salvaged via minor feature modifications.

As illustrated in Figure 10, there are two groups of tests that are made at the feature level. Figure 11 illustrates the procedure for performing the first group of tests that consider only the machine against the feature. The first test is the weight test, which compares the weight of the part (estimated by the product of the part bounding box volume and part material density) with the table weight capacity of the machine. If the machine table weight capacity exceeds the weight of the part then the test is passed. The next test determines if the minimum tolerance specification is greater than or equal to the position accuracy of the machine, if so the machine passes the test. The last test is a tool travel test. Tool travel requirements vary from feature to feature, while tool travel capability varies from machine to machine. For example, cylindrical holes require one axis of travel and drill type machines provide travel along one axis, which is a potential match. Cuboid pockets require three axes of travel unless they are made with an EDM machine. The number of special circumstances involved in tool travel checking makes it difficult to generalize and modularize. The procedure progresses from one-axis travel machines to three-axis travel machines. Rules for different feature types are considered within the context of the travel capabilities of the machine in question. The procedure for checking tool travel will be detailed later.

Figure 12 illustrates the tests that consider the machine and tool simultaneously. The procedure begins by checking if the specified part material is compatible with the cutter by looking for a match between the AISI code of the material and a list of materials assigned to the tool assembly's material cutting capability list. If no match is

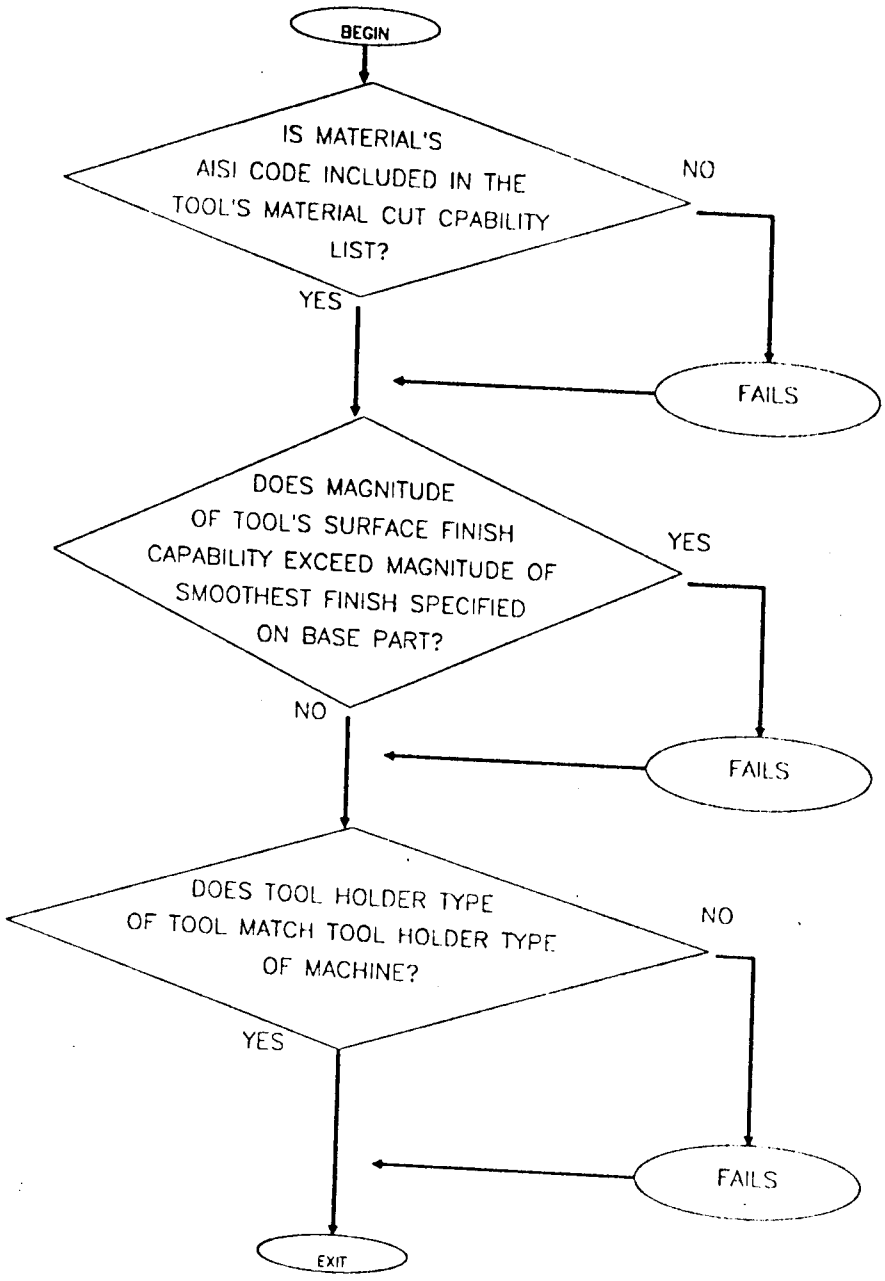


Figure 9: Test Machine and Tool Criteria at the Part Level.

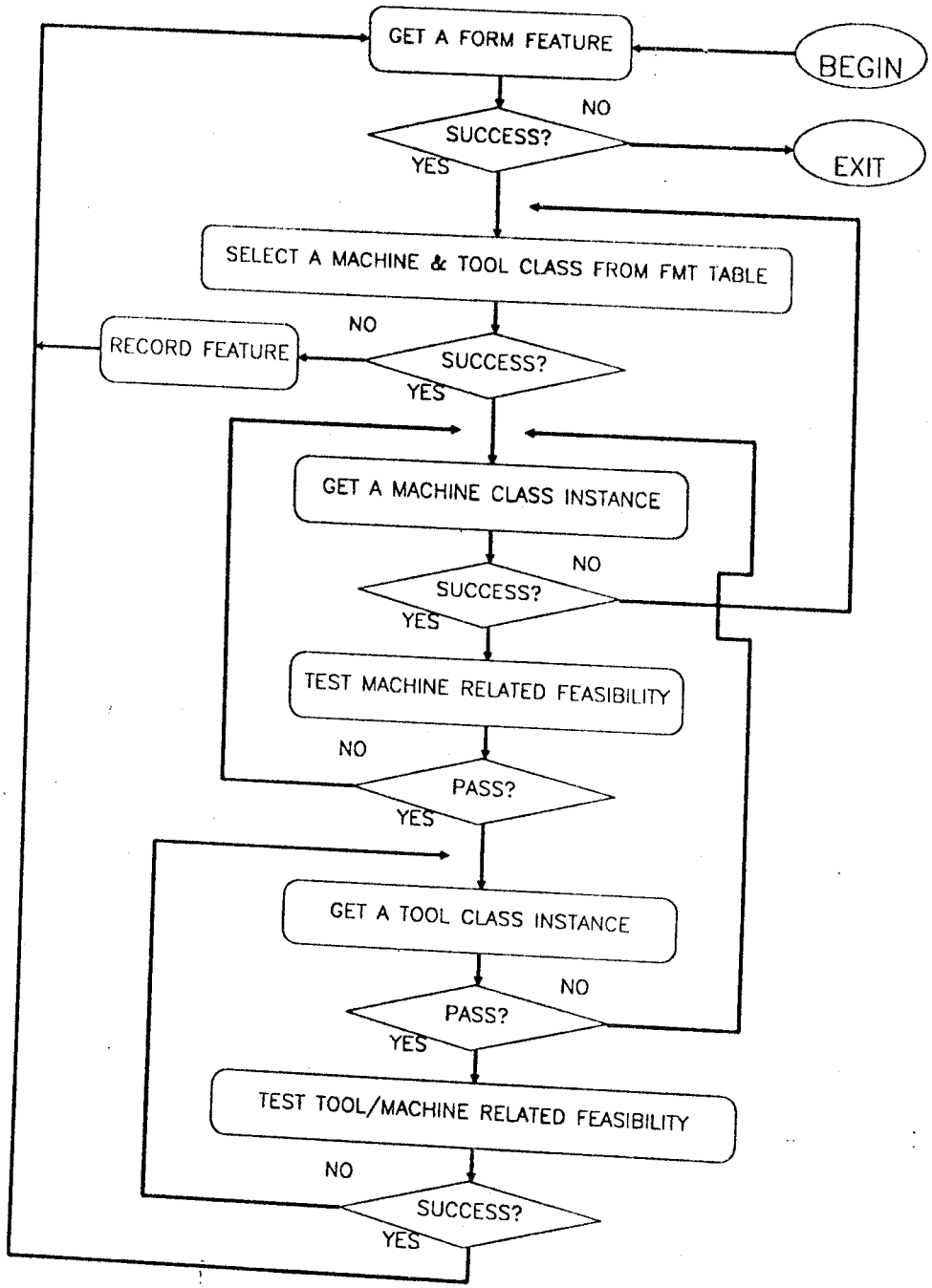


Figure 10: Feature Level Feasibility Assessment.

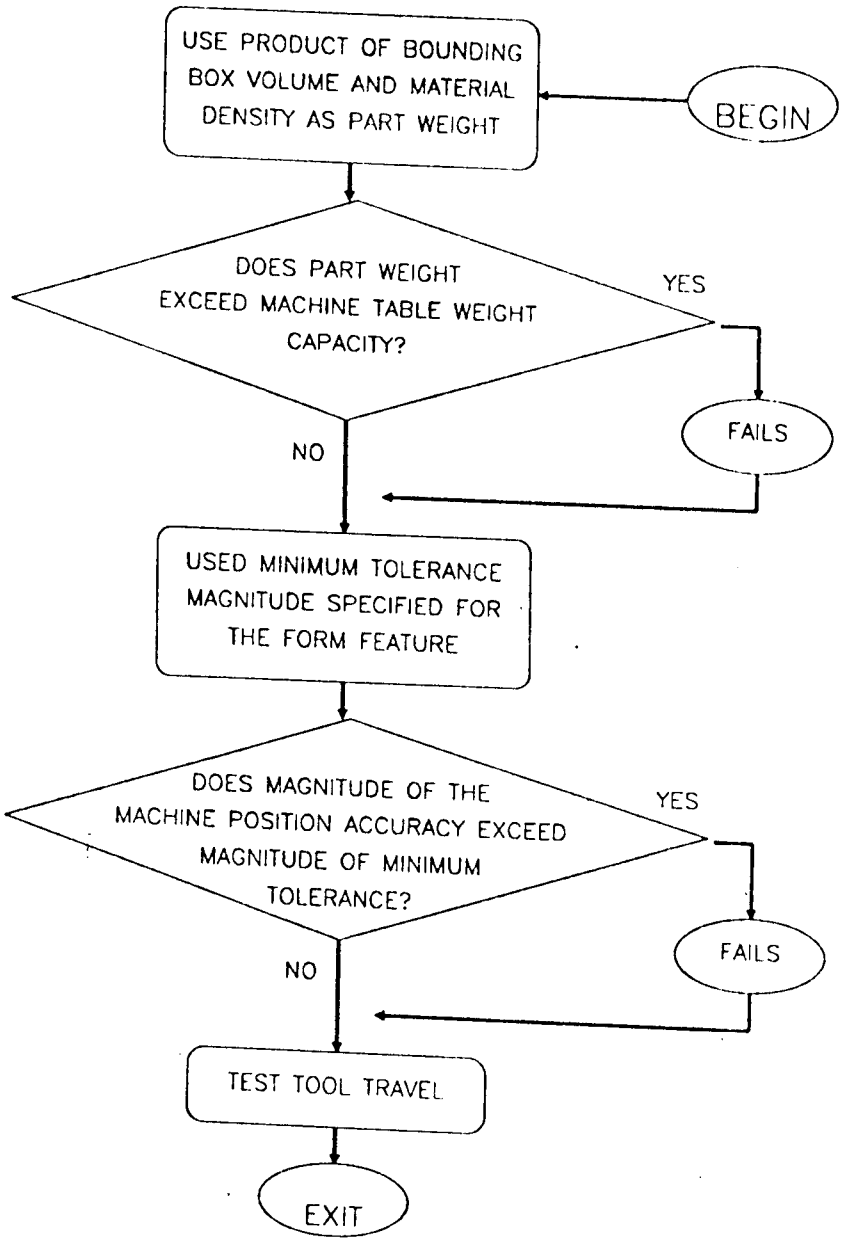


Figure 11: Test Machine Criteria for a Feature.

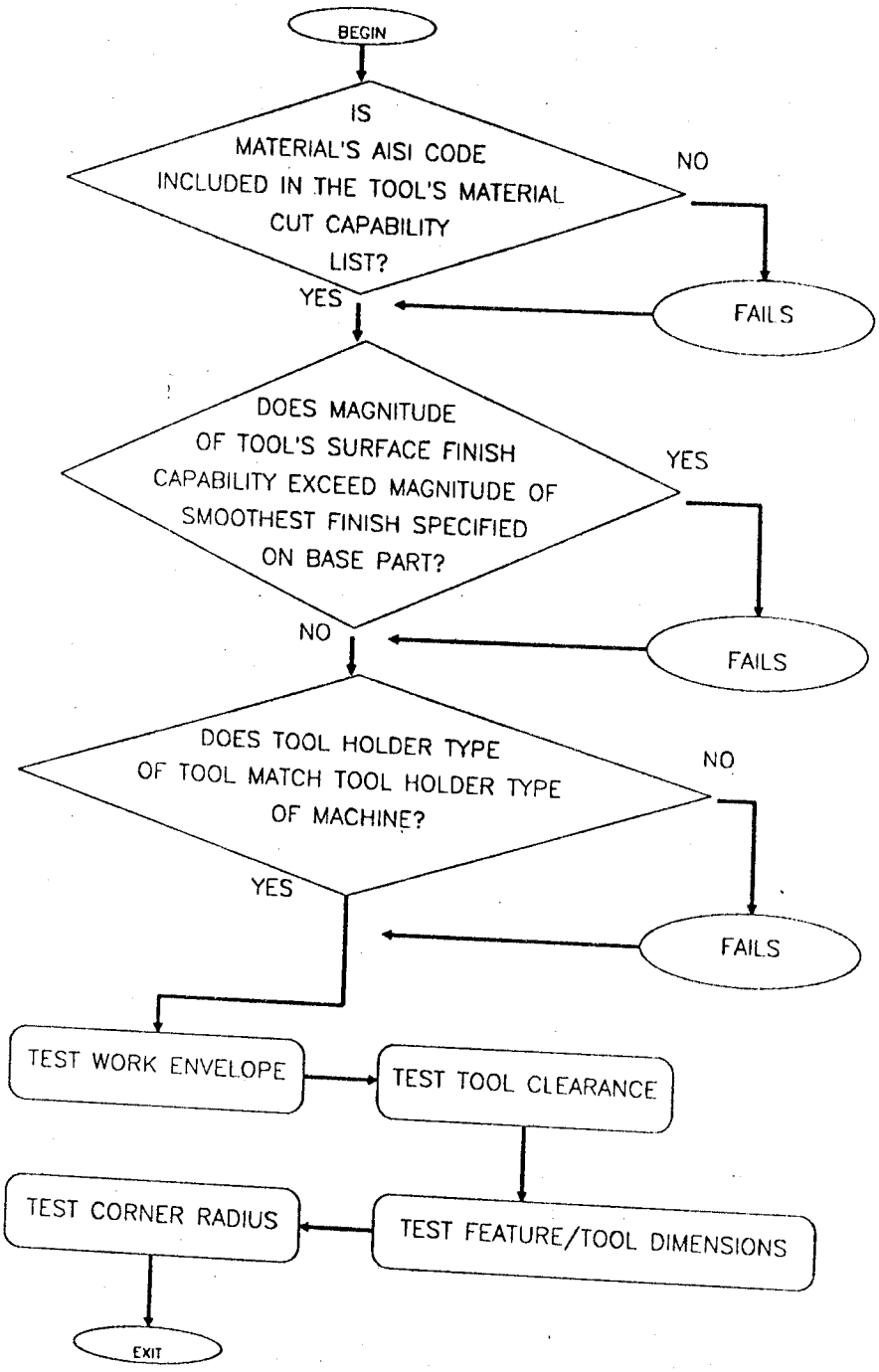


Figure 12: Test Machine and Tool Criteria for a Feature.

found the tool fails the test. Similarly, the surface finish capability of the tool is checked for a match with the minimum surface roughness specified for the feature. The tool fails the test if no match is found. The third test seeks to verify that the tool holder type for the tool matches the tool holder type for the machine instance. No match means the tool fails the test. The next four tests are much more involved and are detailed by their own flow charts, they are; the work envelope test, the tool clearance test, the corner radius test, and the tool/feature dimensions test. These tests will be discussed next, beginning with the work envelope test.

Figure 13 details the procedure for checking the work envelope. Because the work envelope is checked at the part level and at the feature level, the procedure for both levels is presented simultaneously. Therefore, the procedure forks into two branches, one branch for the feature level check and the other for the part level check. The part level check is very simple. The cut-to-length stock dimensions are sorted from maximum to minimum. Then the work envelope dimensions are sorted from maximum to minimum. A conservative test is then made to determine if the minimum dimension of the machine's work envelope is at least as large as the maximum dimension of the part. If not the machine fails. This test is made conservatively so that one may assume that the part can be positioned as necessary during the tool travel test. The feature level test is basically the same, except that the part bounding box dimensions are used instead of the stock dimensions, and the tool assembly length is subtracted from the Z-axis of the machine's work envelope. The Z-axis is considered coincident with the tool axes for the cases listed in Figure 13.

The tool travel test is very complicated, as can be seen from Figure 14. Figure 14 begins by getting and sorting the X, Y, and Z tool travels and the feature dimensions. Note that the Z-axis is assigned to the spindle axis for drills, mills, etc. Once the travel and feature dimensions have been sorted, then the proper rules are applied by considering the type of machine, tool and feature involved. If the machine is any type of drill or electrode EDM the feature dimension along the contact face normal can not exceed the tool travel in the Z-direction. If the machine is a band saw or hack saw then there is a need to differentiate between through-slots and other features that a saw can make. The minimum dimension normal the a contact face (the depth of the slot) can not exceed the tool travel in the Z-direction. For the other feature-types the middle feature dimension is not to exceed the tool travel in the Z-direction. Consider cutting a through-step or a boss with a saw and the objective becomes clear, one must pass the part through the saw in two orientations to cut out a single step. In one orientation the minimum feature dimension is in the direction of travel, while in the other orientation the middle dimension is.

Wire EDM machines cut in a plane that is normal to the depth of a through-hole. For this reason one needs to compare the largest cross-section dimensions of the hole with the X and Y tool travel of the machine. If the largest dimension does not exceed the largest tool travel and the next largest dimension does not exceed the other tool travel, then the machine passes the travel test. If the feature is not a through-hole, then it must be a boss or through-step, because these are the features that a wire EDM can make. If the feature is a boss or through-step, then the middle dimension can not exceed

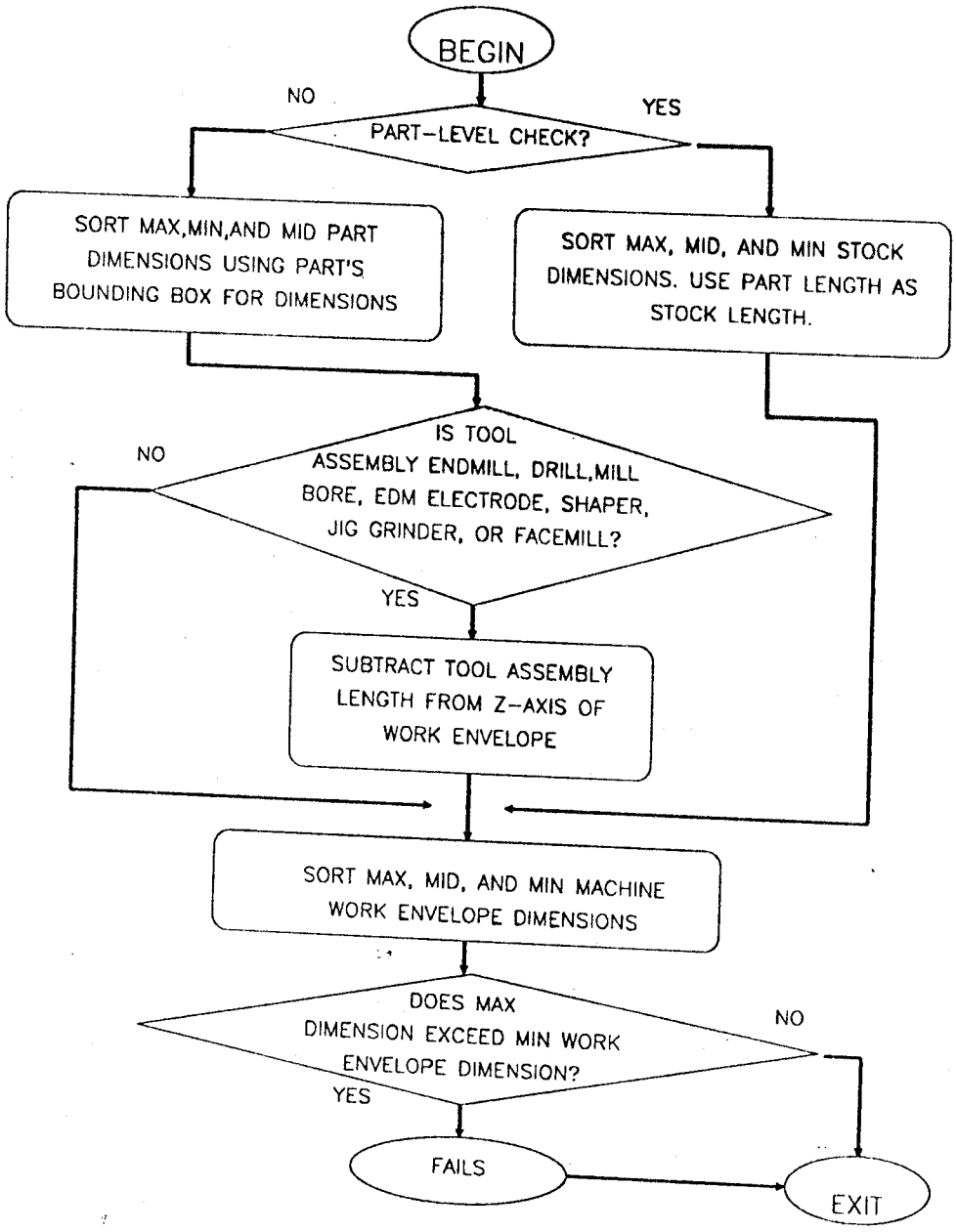


Figure 13: Test Work Envelope.

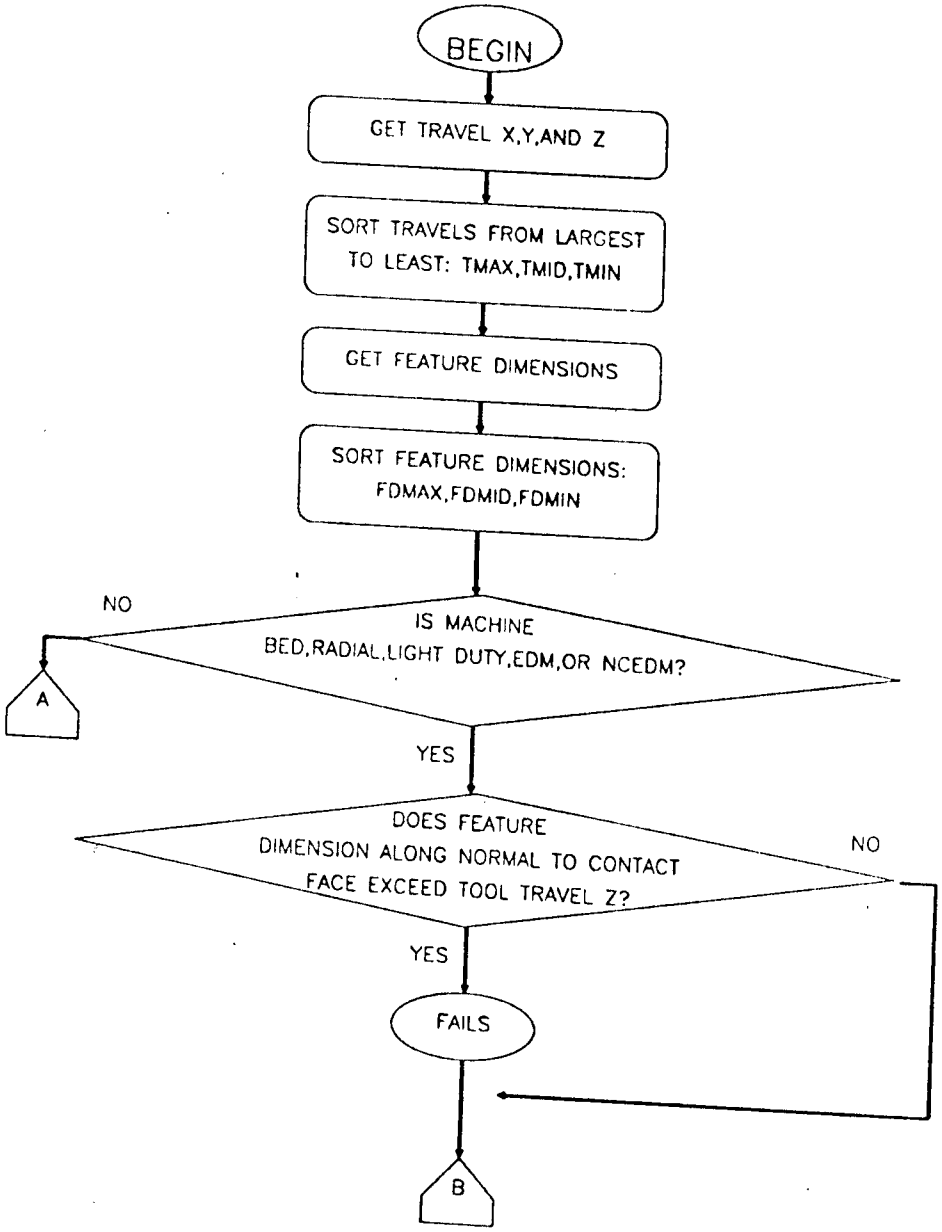


Figure 14: Test Tool Travel.

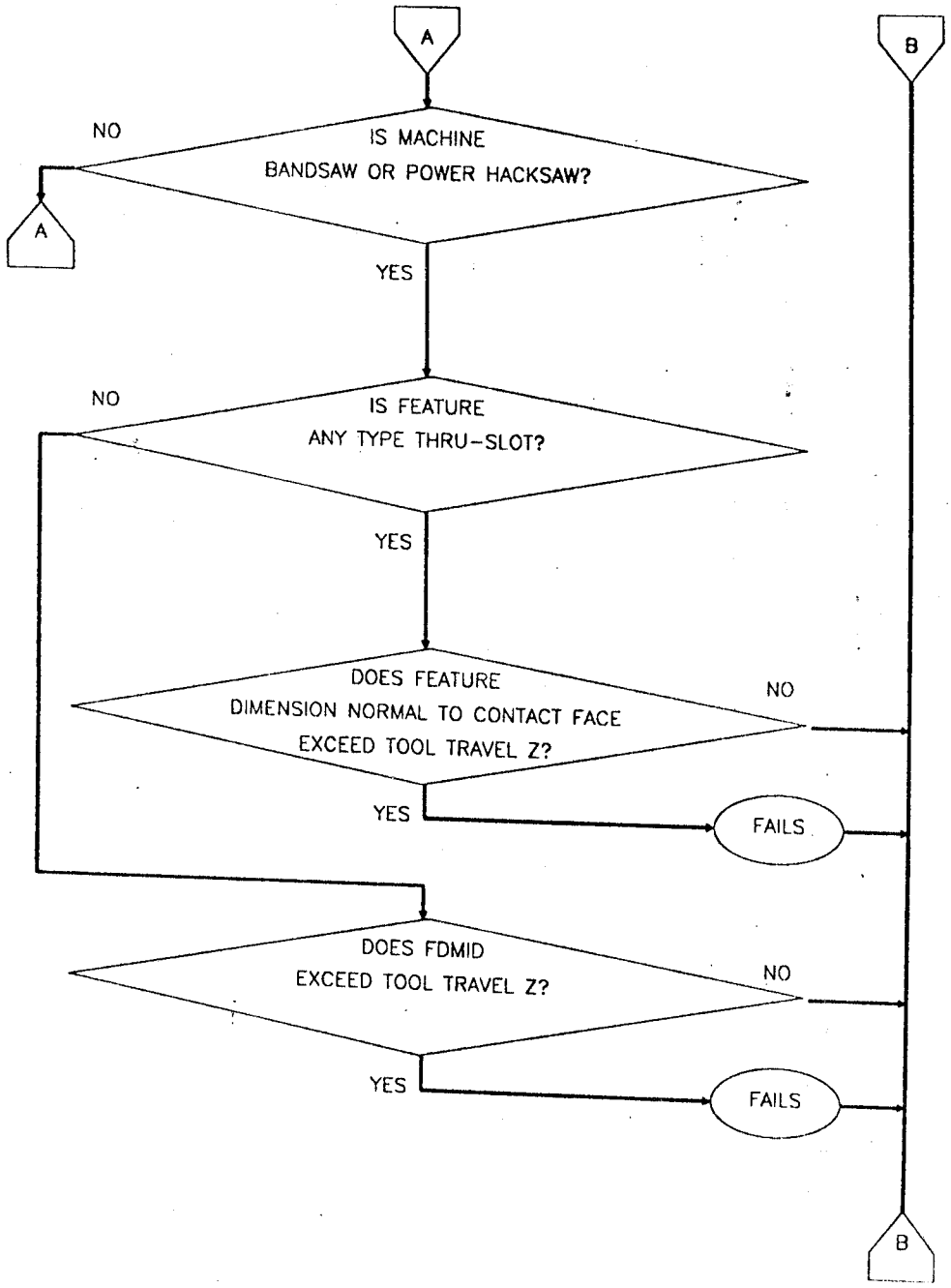


Figure 14: Tool Travel Test (Continued).

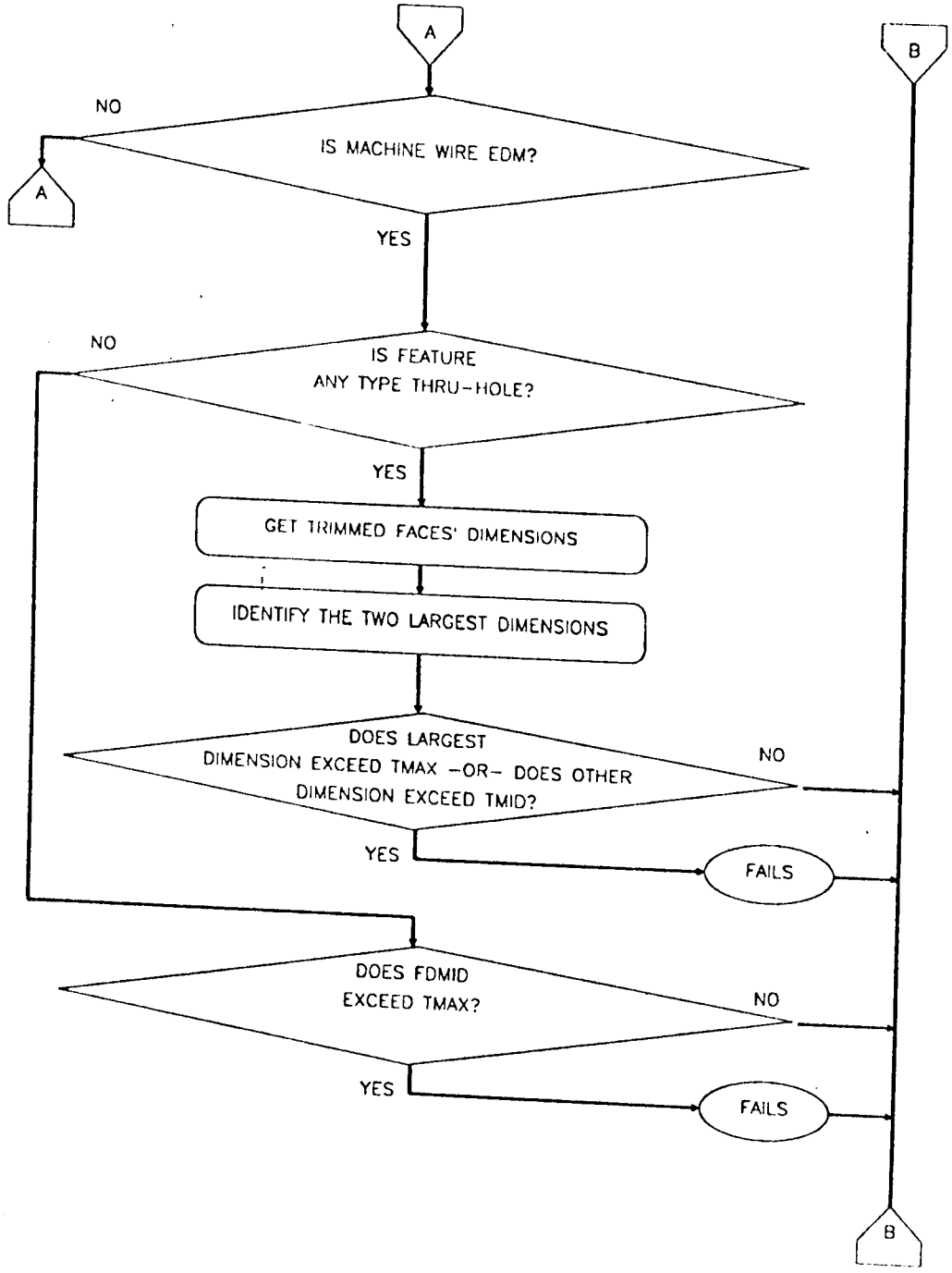


Figure 14: Tool Travel Test (Continued).

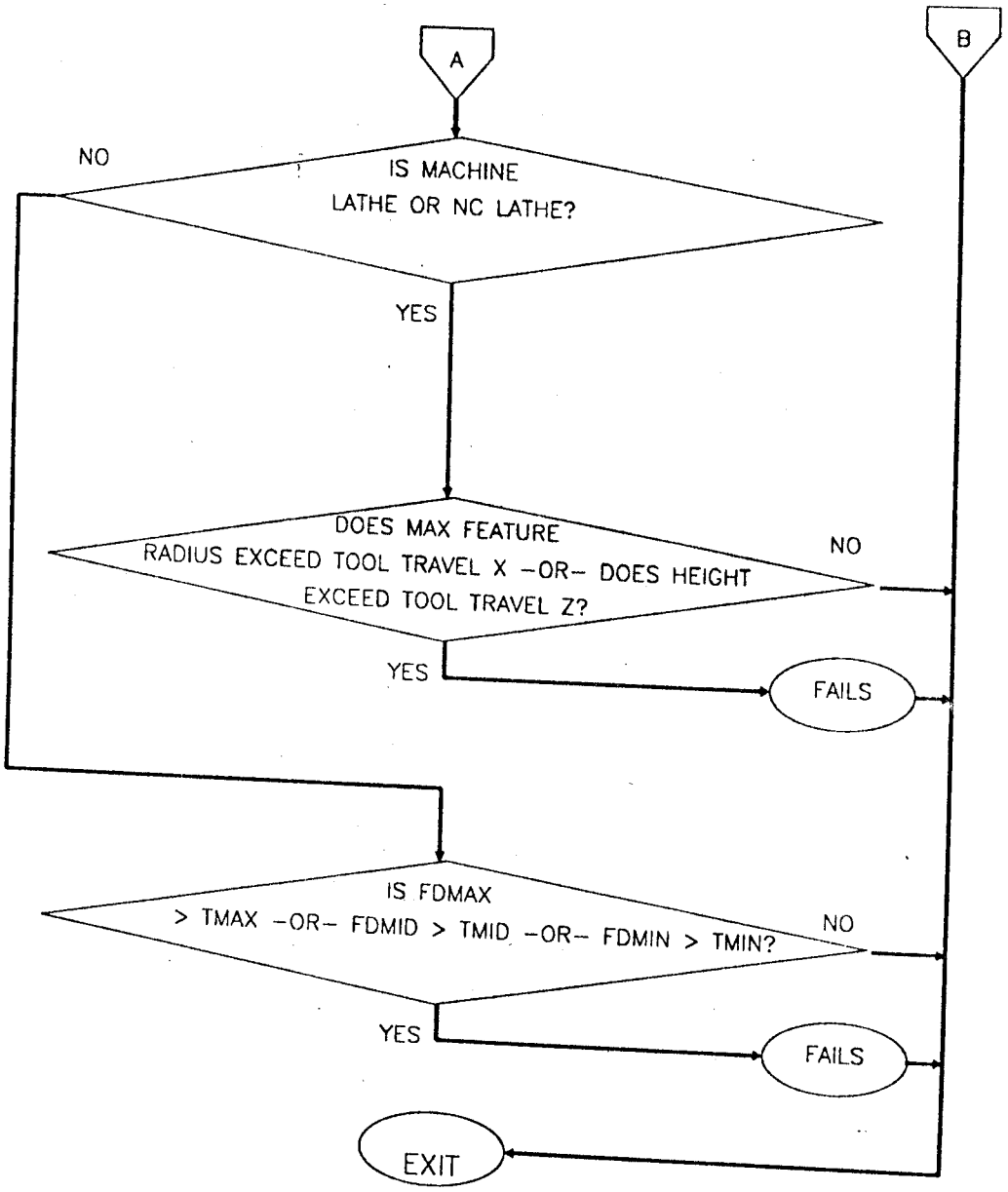


Figure 14: Tool Travel Test (Continued).

the largest tool travel.

Lathes have travel in the Z and X directions and are used to make cylindrical features. The maximum radius of the feature is compared to the tool travel in the X direction, and the height (or depth) of the feature is compared with the travel in the Z direction. Again, the dimensions can not exceed their respective travels. The last group of machines are 3-axis machines, and a direct comparison is made. The largest travel with the largest dimension, the middle dimension with the middle travel, etc.

The corner radius test only considers depression features that have radius corners, and therefore, the first step in the test is to determine if the feature is a radius depression, if not the test is terminated. Please refer to Figure 15. After the feature is identified as a radius depression, the corner length is accessed. The corner length is the distance from the point of tangential intersection of a circle with the minimum possible corner radius and the side of the feature to the vertex that makes the corner. This dimension is illustrated in Figure 16, which will be discussed later. The corner length is used to compute the minimum radius, which changes according to the angle between the two sides that form the corner. The next step is to determine if the feature is any type of radius polygon depression feature, or more specifically, a convex general polygon feature, named for the shape of the cross-section of its primitive solid. If the feature is a radius polygon, then the smallest interior angle on the cross-section is computed by taking the dot product of vectors attached to pairs of sides that form the corners of the polygon. If the tool is a facemill, endmill, millbore, jig grinder or wire EDM the cutter body radius can not exceed the product of the corner length and the tangent of half the minimum interior angle. If the tool is a shaper, then the edge radius can not exceed the same value. If the feature is not a general polygon, then it must be a regular polygon, and the same tests are performed using the product of corner length and tangent of $\pi(N-2)/2N$, where N is the number of sides on the cross-section of the feature primitive. This fork in the procedure represents an attempt to avoid lengthy computations if a simple formula is available. Since EDM electrodes in the database currently do not have an edge radius attribute, they fail the test. If the attribute is added in the future, then the test will be expanded to include EDM electrodes. For clarity, the relationship of minimum cutter radius to corner length is detailed in Figure 16. Note that the corner length and cutter radius form a right triangle, and that the tangent of the half-angle, Phi, is R/C. From this one derives the general relationship. If the polygon is of N equal sides, then Phi becomes $\pi(N-2)/2N$.

The tool clearance check is made between two bosses or between a boss and its parent depression. According to Figure 17, the minimum non-zero clearance is computed between the current feature and another that fits the description above. Tools such as drills, reamers and EDM electrodes may be selected by the FMT table, but are not capable of producing nested features such a bosses in pockets, and are eliminated from consideration. These tools could be selected if the current feature is a depression that can be made by the tool, because when referencing the FMT only the current feature is used to make the selection. Of course, the FMT table does not know that there is another feature inside the depression that renders this tool infeasible, so the procedure must be set up to recognize these conditions. This could be considered a bonus feature

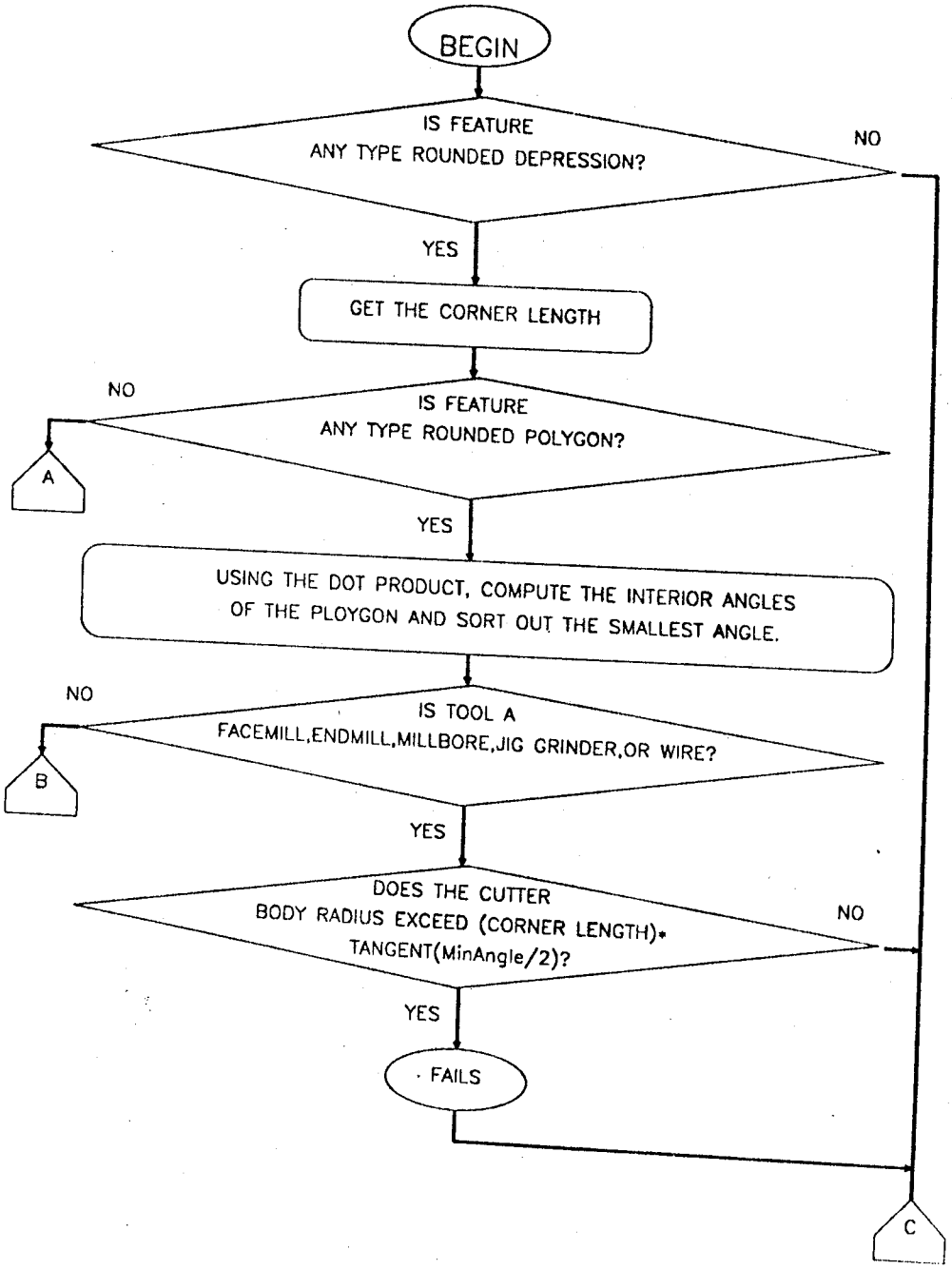


Figure 15: Test Corner Radius.

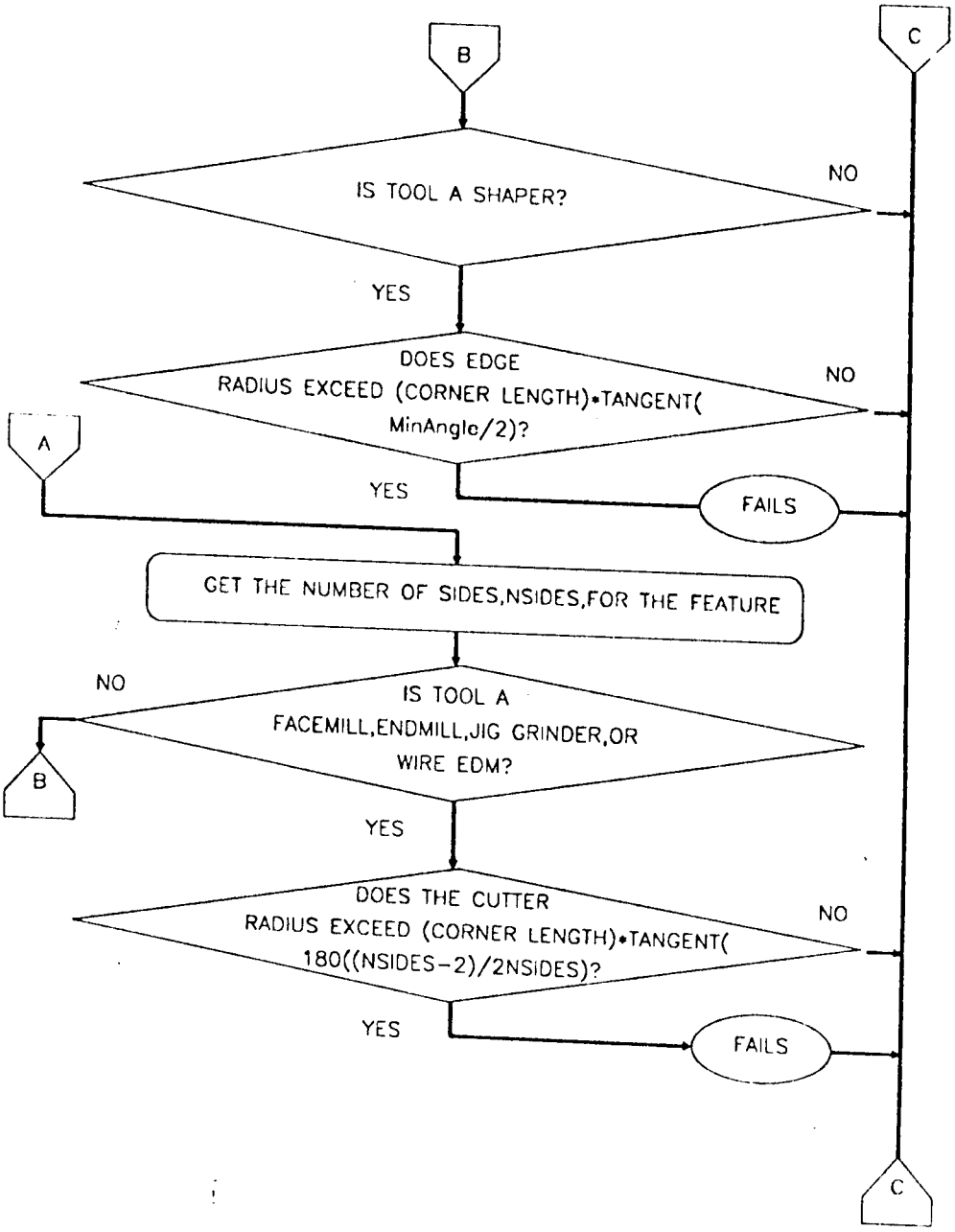


Figure 15: Corner Radius Test (Continued).

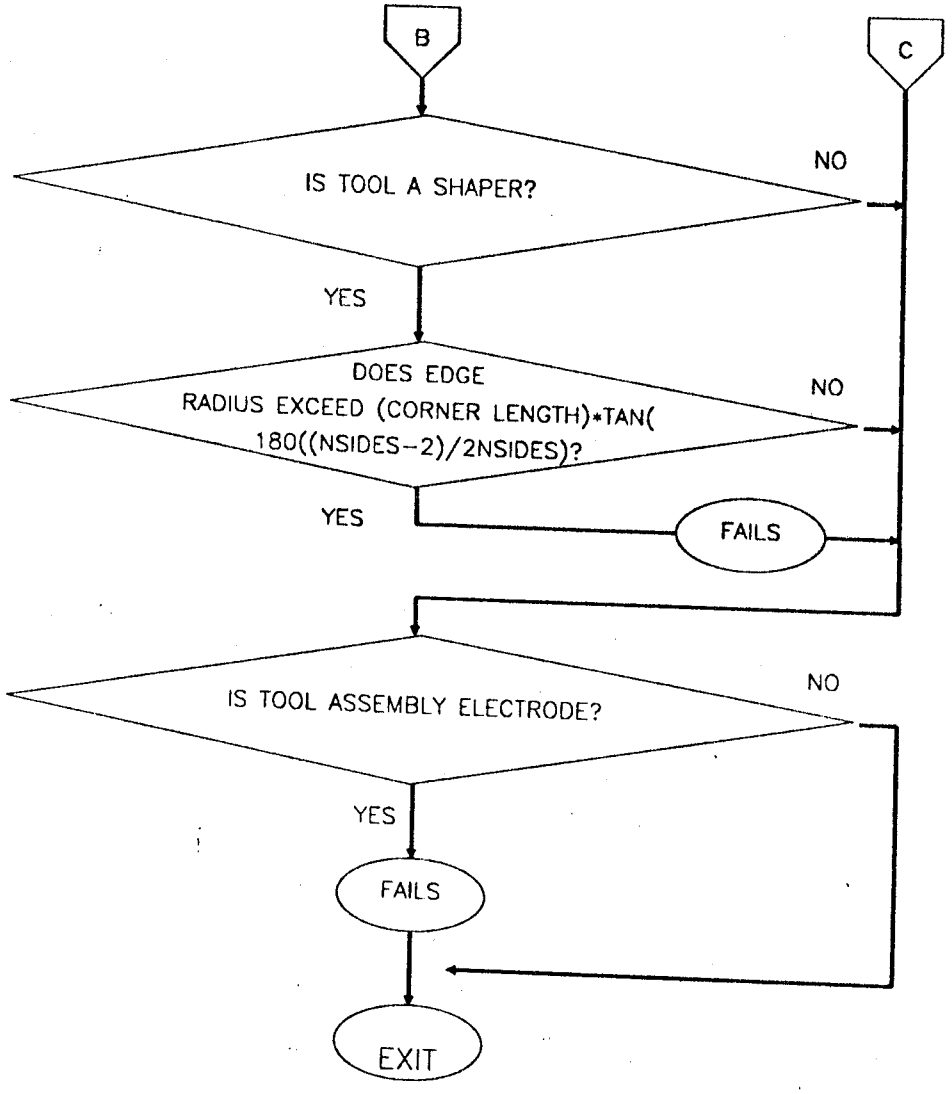


Figure 15: Corner Radius Test (Continued).

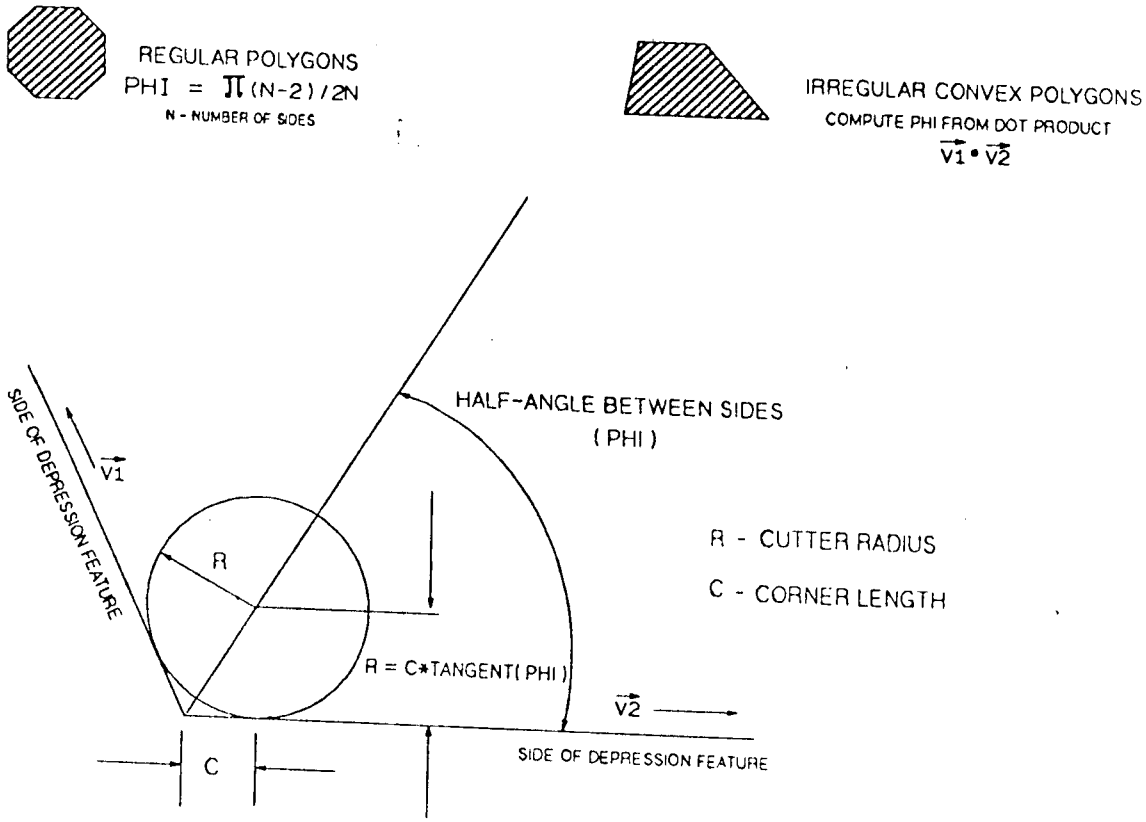


Figure 16: Geometric Representation of Corner Radius Test.

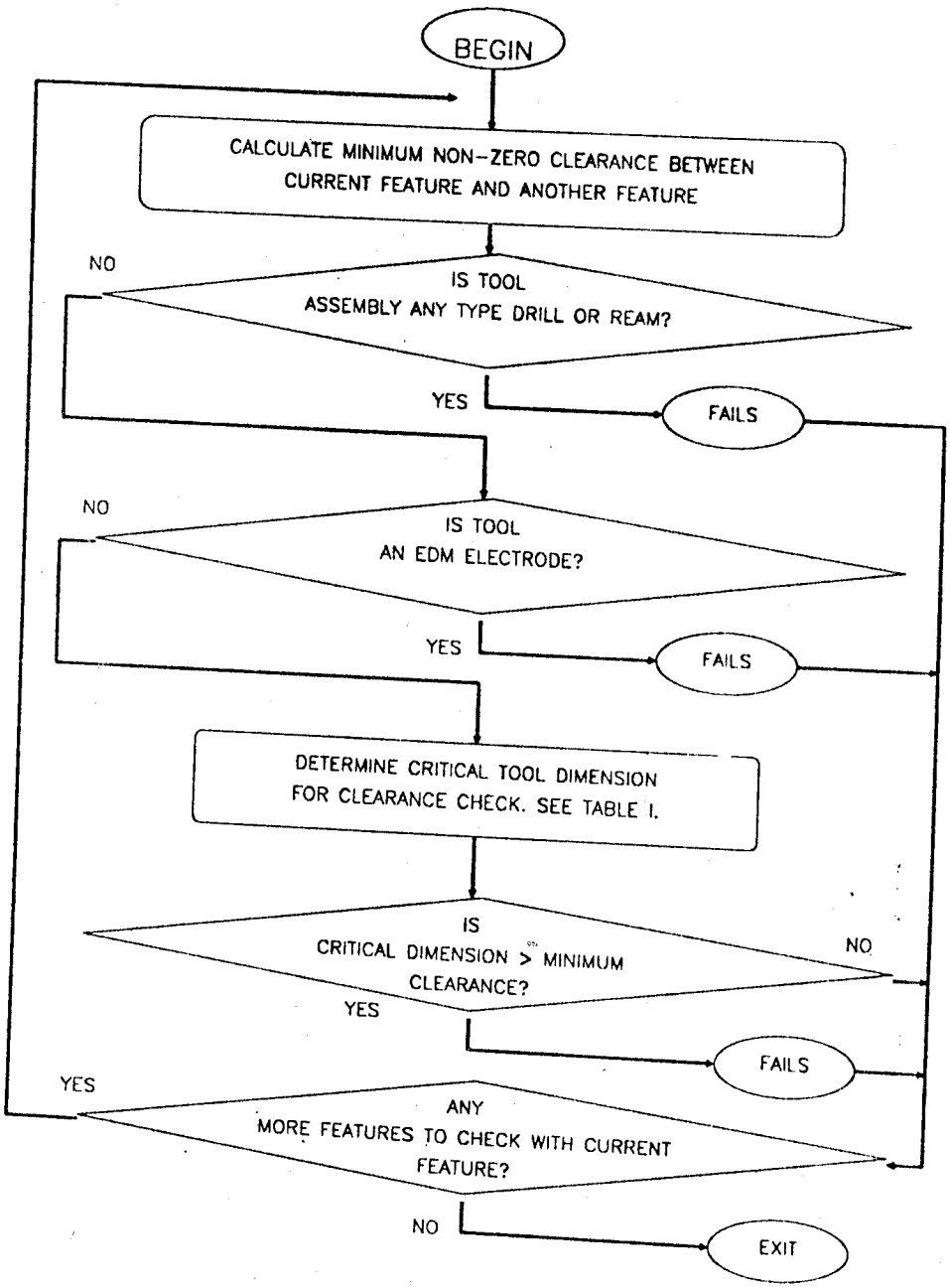


Figure 17: Tool Clearance Test.

of the algorithm, because it disqualifies tools that can not deal with nested features. The vertices of the features' bounding boxes are used for computing clearance, which is done by checking the distances between the points that represent the vertices. Next a critical tool dimension is used for testing the clearance. Table I contains a summary of these critical dimensions for a number of tool classes. If the critical damson is less than or equal to the minimum non-zero clearance between the two features, then the tool passes the clearance test. The critical width of a tool is determined primarily based on how the tool is oriented when cutting. For example, from Table I the endmill's critical dimension is
its cutter body diameter, while the side mill's critical dimension is its cutter body width. Figure 18 illustrates the tool clearance test. Clearance is checked between boss 2 and boss 3, because they are on the same face. Boss 1 is not considered, because it is not on a face that is common with another boss, nor is it within a depression feature. Note that the height difference is ignored when computing the minimum non-zero clearance between features. If the height difference were not ignored, then two bosses of nearly the same height would be assigned a small clearance, even if they are actually widely spaced on the same face. The other feature pair that is checked is the blind slot and boss 4, because the boss is nested inside the slot. Again, if there is a height difference it is ignored.

Besides the tool travel test, the tool dimension test is the most complex in terms of the amount of interaction between feature, machine and tool classes involved. Figure 19 provides general picture of the procedure. If the machine class is EDM then Tool

Table I: Critical Tool Dimension for Clearance of Tool Between Form-Features

Tool Type	Critical Dimension
Surface Grinder	Grinder Body Width
Side Mill	Cutter Body Width
Jig Grinder	Grinder Body Diameter
End & Peripheral Mill	Cutter Body Diameter
Saw	Cutter Body Thickness
Shaper	Cutter Body Thickness
Wire EDM	Cutter Body Diameter
Bore	Minimum Cutter Body Diameter
Single-Point Lathe	Cutter Body Width

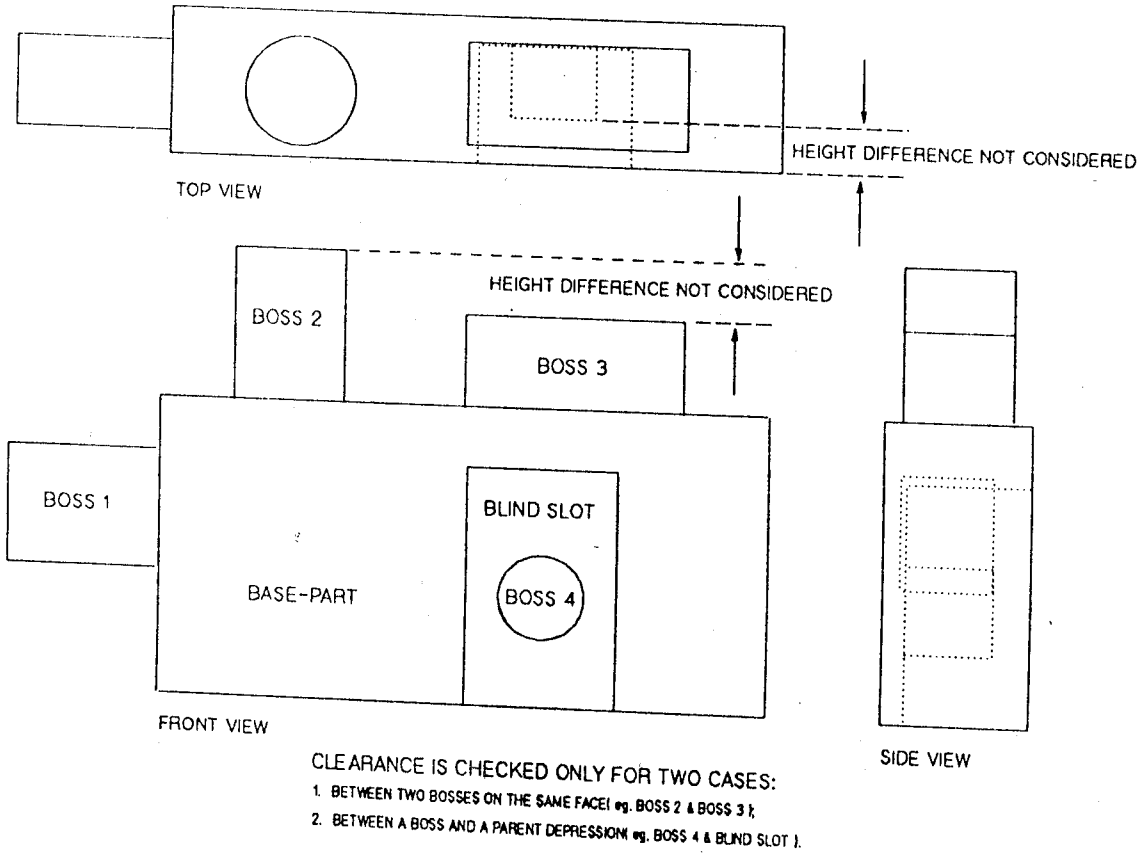


Figure 18: Illustration of the Tool Clearance Test.

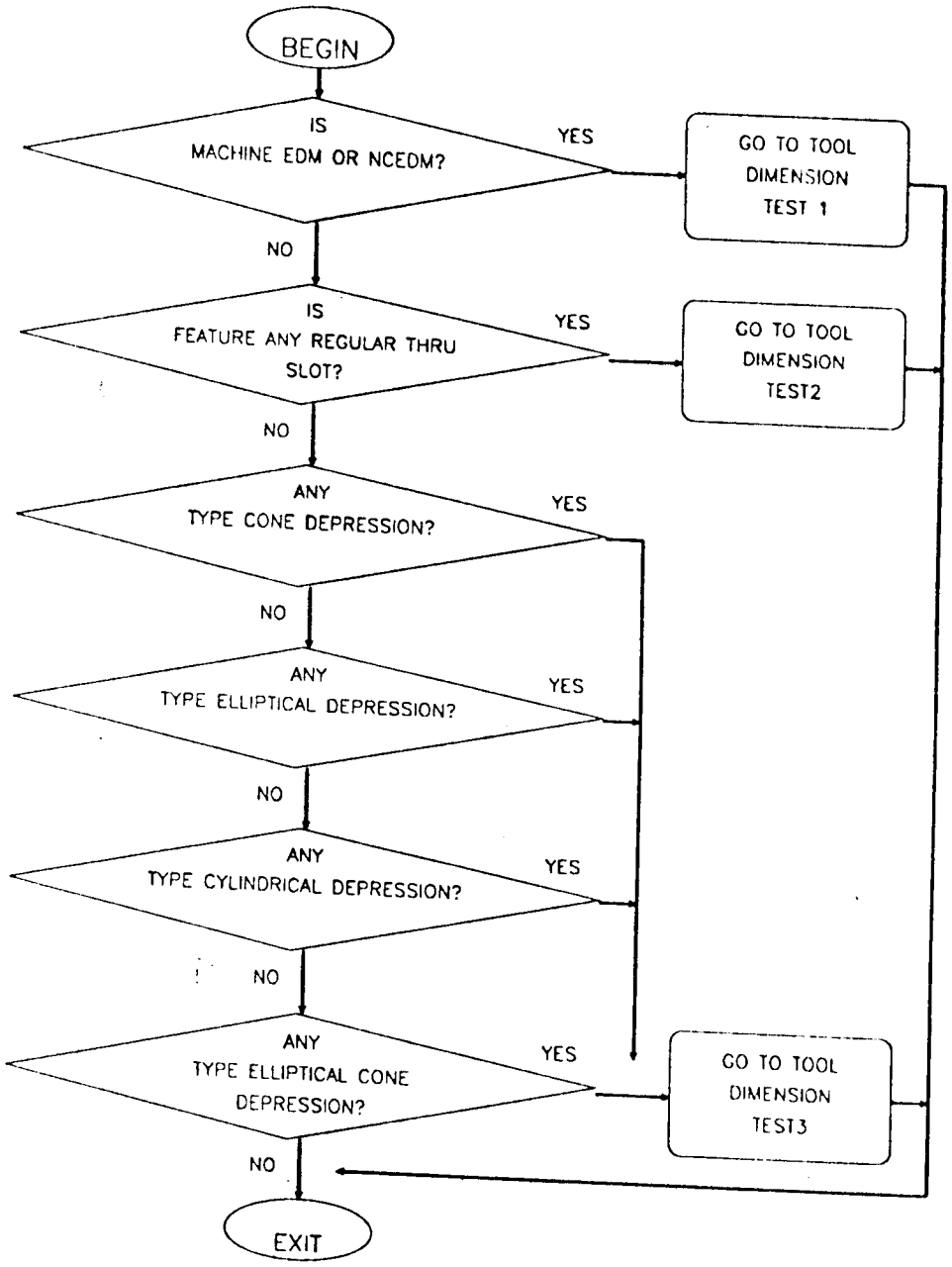


Figure 19: Tool Dimension Test.

Dimension Test 1 is performed. If the feature is any type of square cornered through-slot, then Tool Dimension Test 2 is performed. The last feature-types to be considered are either circular or elliptical. For these feature-types tool dimension test 3 is performed. Depression features with radius corners are not considered here, because the tool is tested against the corner radius, which is smaller than any gross feature dimension. Bosses are not considered, because a tool does not need to be positioned inside the boundaries of a boss for machining.

Figure 20 details Tool Dimension Test 1, which checks if the feature is either a cuboid or cylindrical depression. Only these two feature-types can be made from the stock of electrodes currently in the database. If the cross-section of the electrode perfectly matches that of the feature, and if the length of the electrode (depth in the tool attribute list) is at least as large as the depth of the feature, then the tool passes the test. If the feature-type is not a cylindrical or cuboid depression the tool fails the test.

Figure 21 illustrates Tool Dimension Test 2, which is for through-slots. The first step in this procedure is to determine the critical width of the slot. This dimension is then compared with the critical dimension of the tool. If the critical tool dimension is larger than the critical slot dimension the tool fails the test. For surface grinders and shapers the critical dimension is the cutter body width, and a second critical dimension, the cutter body radius, is compared with the slot depth for the case of the surface grinder. Cutter body diameter is checked against the critical width for mill, jig grinder and wire EDM tools, unless it is a side mill. For side mills the cutter width is compared with the critical width and the cutter body radius is compared with the depth of the feature. If the tool is a saw, then the test is made between the length of the blade and the slot length. The critical widths are summarized in Table II.

Tool Dimension Test 3 is outlined in Figure 22. This test deals with depression features that have a primitive with either a circular or elliptical cross-section. Drills and reamers must have the same diameter as circular holes and their cutter body length must be at least as long as the depth of the hole. Mills, jig grinders and wire EDM tools must have a diameter that less than or equal to the diameter of the hole. The width of a single point lathe tool can not exceed the minimum diameter of the feature. For bores, the minimum cutter diameter can not exceed the minimum feature diameter, nor can the feature depth exceed the cutter body length. Boring bars tend to have large heads that limit tool penetration to the cutter body length. In all cases, if the feature has more then one size radius the minimum radius is used for comparison. For example, cone depressions have a top and a bottom radius due to taper, and ellipses have a major and a minor radius because of their eccentricity.

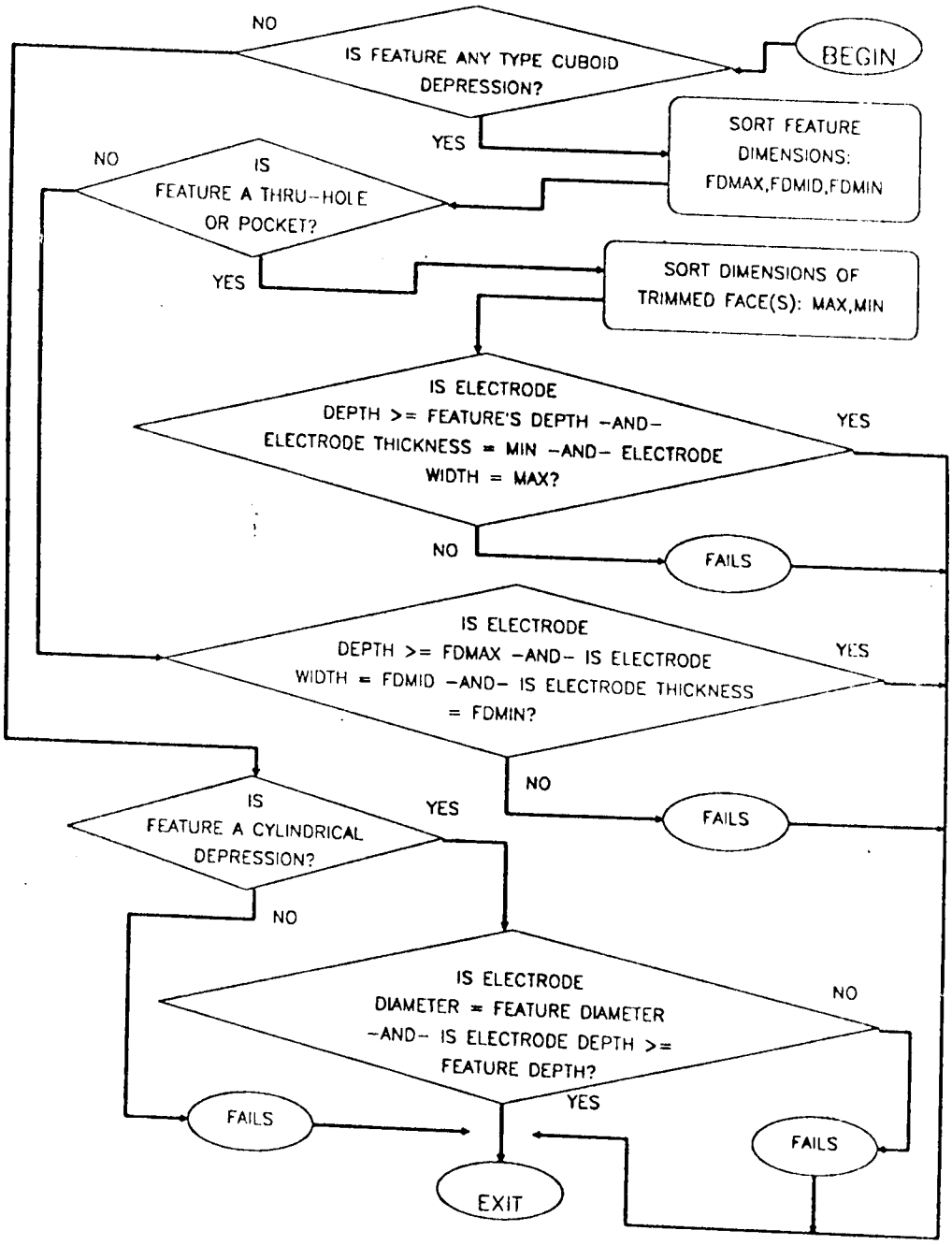


Figure 20: Tool Dimension Test 1.

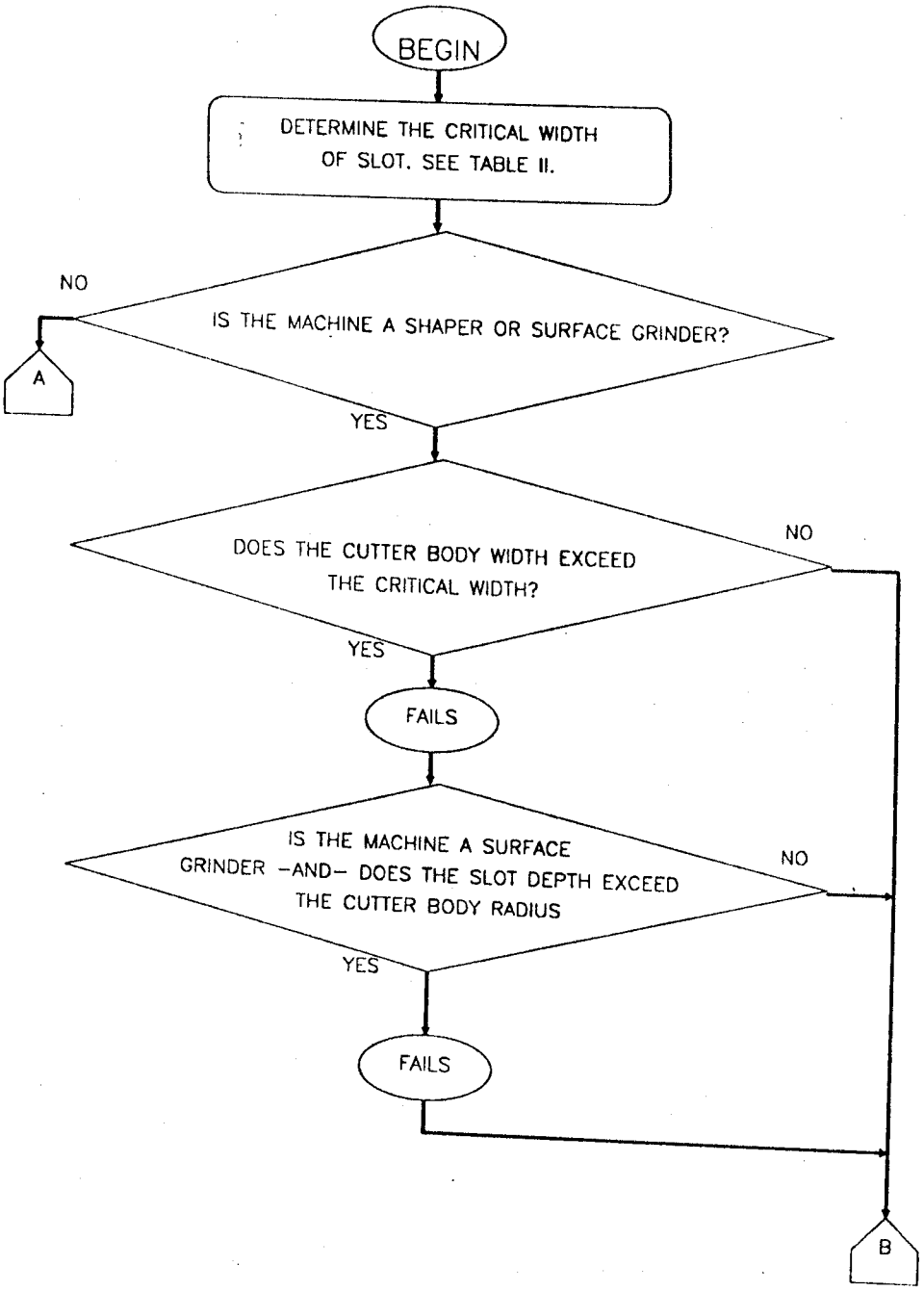


Figure 21: Tool Dimension Test 2.

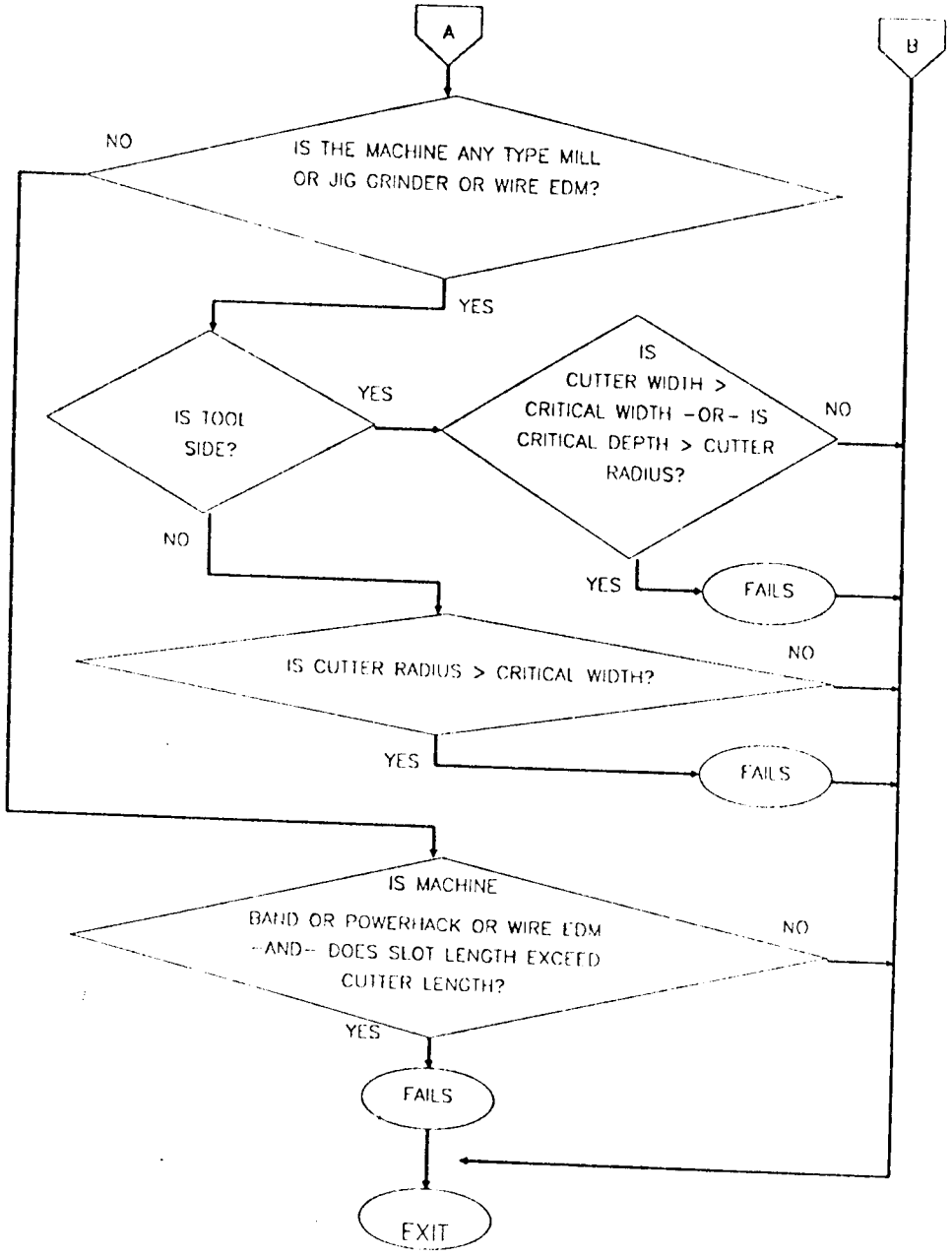


Figure 21: Tool Dimension Test 2 (Continued).

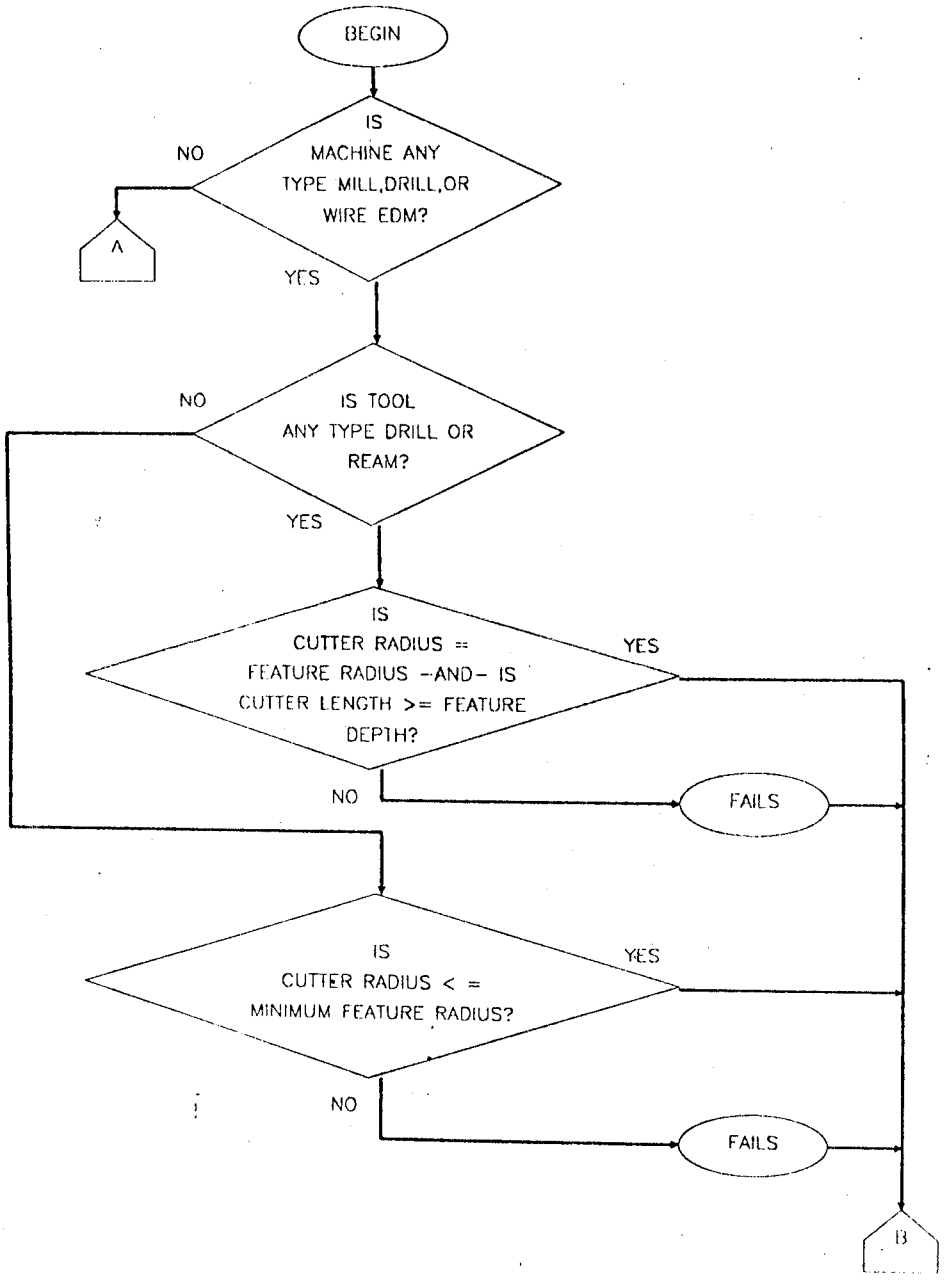


Figure 22: Tool Dimension Test 3.

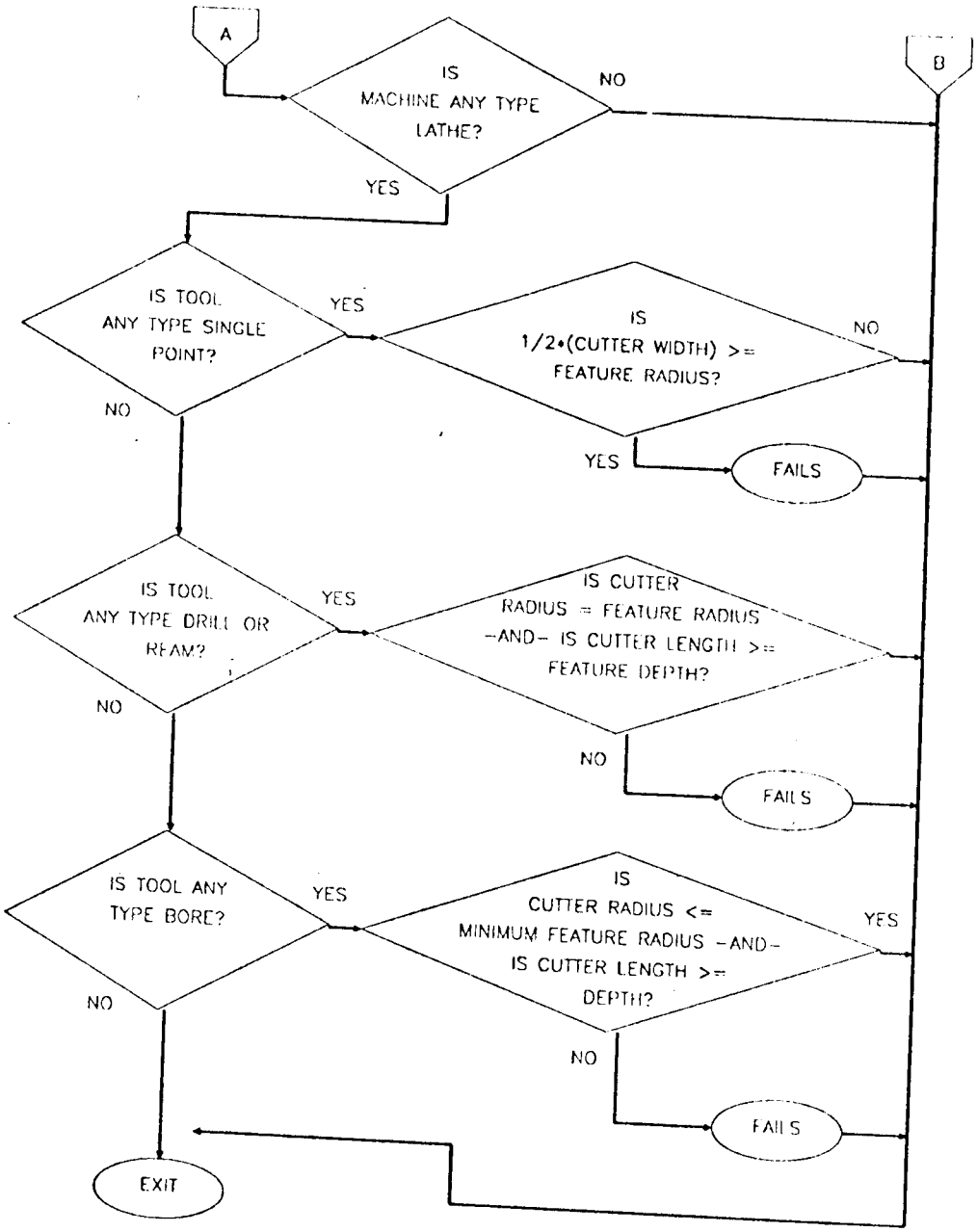


Figure 22: Tool Dimension Test 3 (Continued).

Table II: Critical Width Specification for a Given Type of Thru-Slot

Type of Slot	Critical Width
Regular_Cuboid_Thru_Slot	Minimum Dimension Parallel to Contact Faces
Regular_N_Sided_Thru_Slot	Length Of One Side Of Primitive Cross-Section
Regular_CC_Thru_Slot	Length Of One Side Of Primitive Cross-Section
Regular_Wedged_Thru_Slot	Length Of One Side Of Primitive Cross-Section
Regular_P_Round_Thru_Slot	Length Of Smallest Side Of Primitive Cross-Section
Regular_Diamond_Thru_Slot	Length Of Smallest Side Of Primitive Cross-Section
Regular_Trapezoid_Thru_Slot	Length Of Smallest Side Of Primitive Cross-Section

D2.3 Form-Feature Interference Checking

Part feature interference is any instance where any feature(s) on a part preclude(s) the processing of any other feature(s) on the part for any reason. This interference blocks the cutter's access to a feature, thereby rendering the feature infeasible. Such a situation requires redesign of the part, because if one feature on a part is not producible, then the whole part is not. This is different from the equipment capability problem, because the purchase of a new machine-tool or cutter may solve the problem admittedly, at an appreciable cost. The position and orientation of a given feature, as well as its immediate environment (ie. surrounding features and fixture elements), strongly influence the tool path.

Feature interference is tested for by projecting solid primitives through the volume of the product's bounding box, and then checking for intersections of the part with the solids. These primitives are identical in cross-section to features on the part, and each primitive projects from its prototype feature with the same orientation. Please refer to Figure 23. Depression features have a number of projections equal to the number of trimmed faces on the feature. Protrusion features have one projection, and it has the same cross-section as the face in contact with the rest of the part. If all projections from a feature intersect with the part, then there is interference. The part is assessed to be infeasible. Recall from the literature review that this technique was implemented by Joshi et al[24] for checking tool approach directions for cylindrical holes. Referring to Figure 23, the projections are illustrated for boss 2 and for the though-slot. The projection for boss 1 is omitted for clarity. The algorithm for testing for form-feature

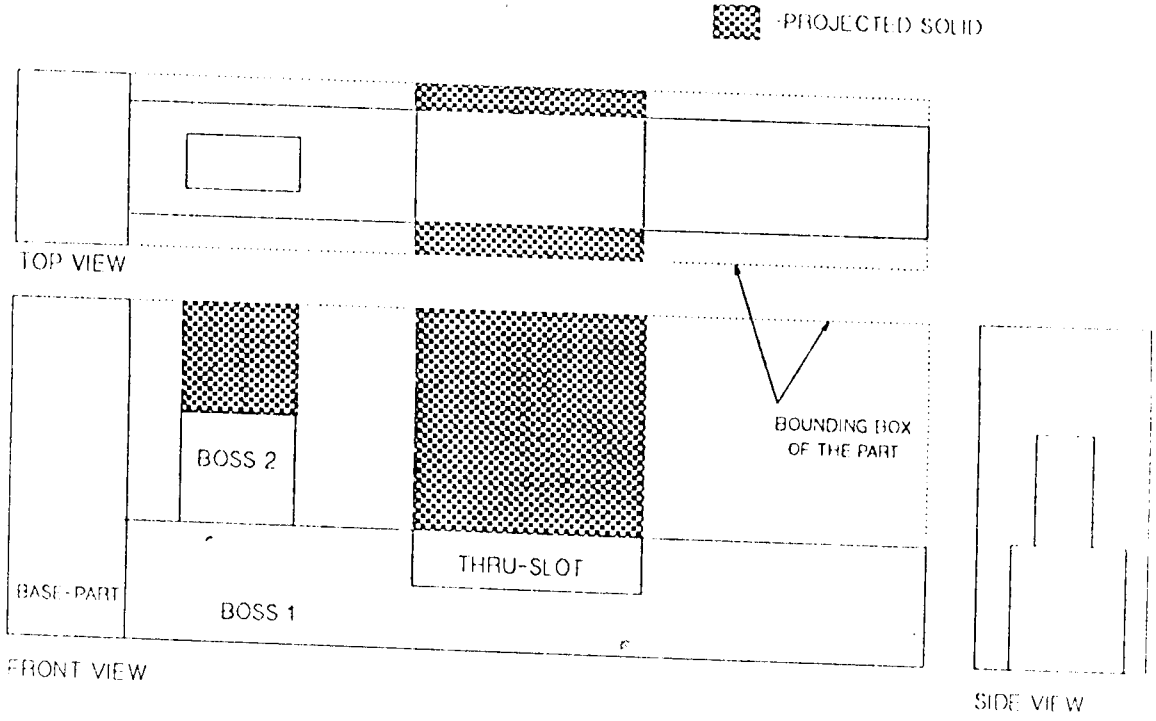


Figure 23: Illustration of Form Feature Interference Test.

interference is not implemented for this thesis, but the general procedure is outlined in Figures 24 and 25. Referring to Figure 24, the first step is to get a feature for testing, then the actual test is performed. After the test is finished get the ID of the next feature to be tested. This is done so that an unaltered version of the product model can be retrieved. An unaltered version is retrieved for each test to prevent cluttering the model with projections. After the model has been regenerated the test is made on the feature with the ID that was saved. All features are tested. For the case of a boss, there is only one projection which is subtracted from the part model, and interference is present if any new faces are produced. For depression features, projections are made for each consecutive trimmed face, one face at a time, until all have been projected. It is necessary to project all trimmed faces, because all tool approach paths must be blocked to have interference.

D3. Producibility Index

The producibility index is based on the concept of manufacturing difficulty. This concept allows one to focus attention on a narrow, yet important, set of criteria. Recall that Sanchez [4] identified the most important factors affecting the producibility of a given product to be material selection, manufacturing process selection, design geometry, design specifications, and production quantity. Based on this guideline, one can extract difficulty primarily from material selection, design geometry and design specifications. Because this development is focused on the machining process, the difficulty associated with process selection is limited to scrutiny of the various machining processes. Handling and setup difficulty, and fabricating difficulty are considered to be affected by the previously stated parameters and form the basis of the producibility evaluation strategy. The producibility index captures difficulty at the part level and at the feature level by first capturing information of a global nature, and then by computing a difficulty based penalty rating for each form-feature. The two levels of information are necessary for providing a complete assessment of the producibility of a part design. Several parameters are used for computing the producibility index, and are discussed in the following sections.

The selected parameters are then used as a basis for computing an index that captures the effects of all the parameters. This is a composite index and, by capturing inherent difficulty at both the part level and feature level, it is comprehensive within the boundaries of its application. Equation 1 is used to compute the composite index, and is structured to realistically reflect the effects of the producibility parameters. Further explanation will be provided when necessary.

$$P = \prod_{i=1}^{4} p_i \qquad (1)$$

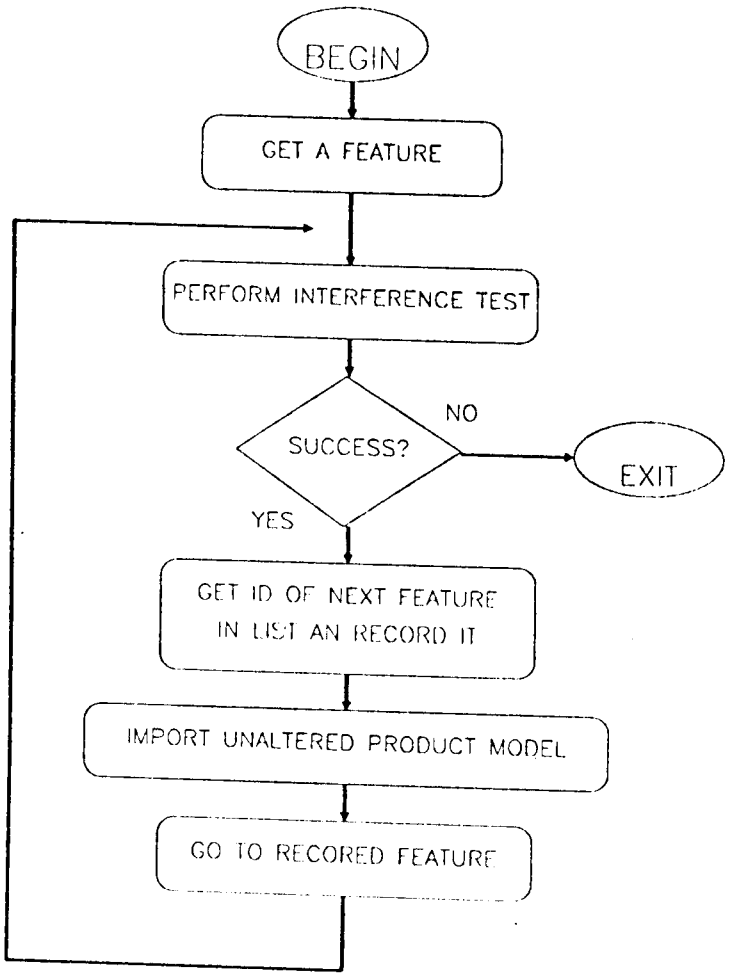


Figure 24: Procedure for Checking Feature Interference.

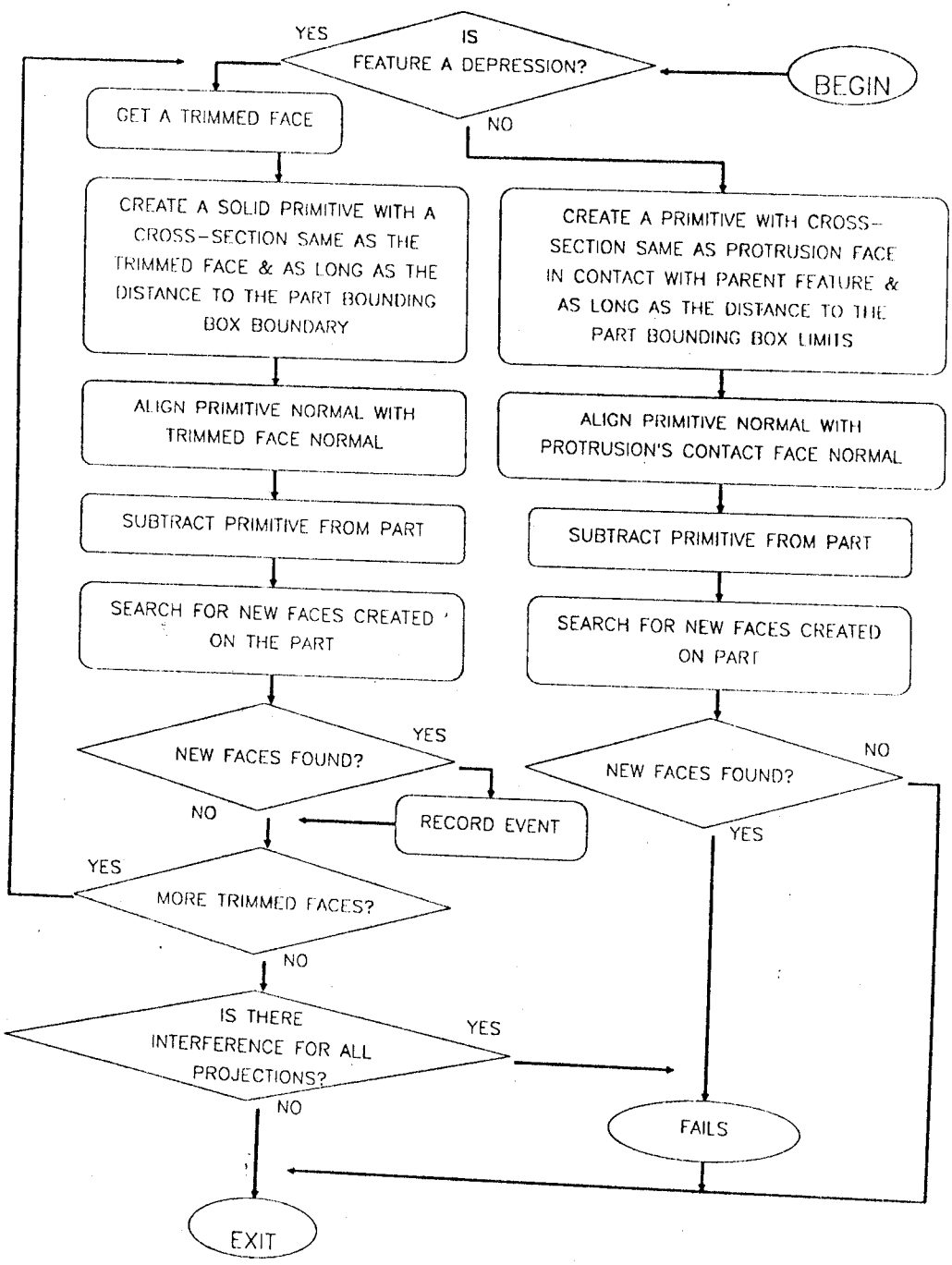


Figure 25: Perform the Interference Test.

Where P is the total producibility index, p_1 is the producibility index for material selection, p_2 is the producibility index for part size, p_3 is the producibility index for part weight and p_4 is the producibility index for cumulative feature effects.

The purpose of equation 1 is to account for the compounding effects of the material selection, part size and weight, and feature characteristics. The combined effect of product level and feature level producibility must be expressed as a product, because the features tend to be dependent upon the product level factors (eg. material selection), and because the global parameters have additive rather than average effects. Development of the four parameter indices is detailed next.

D3.1 Contribution of Material Selection

Material selection is a very important aspect of producibility evaluation for the machining process. Sanchez [4] identified material selection as one of the primary factors affecting producibility. Two important aspects of material selection are material machinability and material availability. Material availability affects cost and schedule compatibility, but is not an indicator of fabricating difficulty. A very useful concept, machinability is used to rate the machining ease of a given material. Features that are easy to produce in one type of material may be very difficult to produce in another, which is to say that materials with a relatively low machinability rating, denoted MR, contribute negatively to producibility. In order to capture the difficulty of a machined component design with respect to machinability, one may wish to create some type of reference value that would contrast with the actual machinability of the specified material. By quantifying the disparity between the reference standard and the actual design specified value, one may be able to normalize the machinability based relative design difficulty. The obvious reference standard for machinability is a machinability rating, either the lowest known value or the highest known value. The highest known machinability value is selected as the reference for this development, and is the machinability rating for free-cutting brass, with a rating of 3.0 [5].

Equation 2 has been developed to capture the producibility of a part design with respect to material selection as follows:

$$p_1 = \min\left(\frac{MR}{MR^*} + 0.5, 1\right) \tag{2}$$

Where MR is the machinability rating for the specified part material and MR^* is the highest known rating. The producibility index for material selection will always be evaluated between 1 and 0.5 due to the author's opinion that aluminum should contribute to a producibility index equal to 1. Aluminum is easily machined (MR ranges from about 1.5 to 2.0) and should not be penalized by the index.

D3.2 Contribution of Part Size

Part size is taken as a geometric aspect of producibility, as described by Sanchez [4]. Form features associated with a small part will tend to be small, and small depression features are likely to be difficult due to accessibility constraints. Clearances

will tend to be tight. Fixturing a very small part is difficult because most fixturing elements are large compared to the part, which may result in excessive setups due to blocked features. The actual size of a part, let it be denoted S_a, is calculated from the dimensions of the smallest cuboid volume that can enclose the part(its bounding box), and is defined as the sum of the height, width and depth of the bounding box. The threshold size, let it be denoted S_t, serves as a reference size for relative handling difficulty with respect to size. Difficulty is then captured in terms of the disparity between the actual size and the threshold size. The producibility with respect to part size is captured in the form of equation 3 as follows:

$$p_2 = 1 - \max\left(\frac{S_t - S_a}{S_t}, 0\right) \tag{3}$$

where S_t is the threshold value for part size and S_a is the actual part size.

D3.3 Contribution of Part Weight

Part weight can be considered to be a result of design geometry and material selection, because the part size and material density determine the weight. Part weight is an important parameter because it can affect material handling and setup ease. Again, a reference is needed. In this case one may wish to capture difficulty relative to machine capacity, as well as human capacity. Machine capacity may be used as the limiting factor, while the human capacity is used as the triggering value. If a part weighs less than the human threshold value, then no difficulty is perceived. When the weight of a part, let it be denoted W_a, is between the maximum machine tool load capacity, let it be denoted W_{Max}, and the capacity of the loading agent, let it be denoted W_t, then there is difficulty with regard to handling the part. The difficulty is captured in terms of the relationship between the actual part weight, the threshold weight and the maximum possible weight. The part design's producibility with respect to weight is capture by equation 4 as follows:

$$p_3 = 1 - \max\left(\frac{W_a - W_t}{W_{Max}}, 0\right) \tag{4}$$

where W_a is the actual weight of the part, W_t is the threshold weight for a part and W_{Max} is the maximum feasible weight for a Part.

D3.4 Contribution of Form Features

Certain form features are inherently more difficult to fabricate than are others -for example, compare the machining requirements for a cylindrical thru-hole with those for a cuboid pocket with sharp corners. Feature attributes such as tolerance and surface finish, and the clearance between features impact heavily on producibility, as does

feature orientation to a lesser degree.

Cumulative effects of all form features are represented by equation 5 as follows:

$$p_4 = b^q; (0<b<1); (q \geq 0) \tag{5}$$

$$q = \sum_{j=0}^{N} f_j \tag{6}$$

where b is a constant for controlling the spread of equation 5 and q is the cumulative penalty. The establishment of a suitable value for b is discussed later in the thesis. The variable f_j is the penalty assigned to jth feature and N is the number of features. If j = N, then the penalty is for minimum clearance between any two features. The clearance penalty is discussed in a later section.

The feature producibility is inversely related to a cumulative penalty that considers the effect of each feature. Feature difficulty can be traced to a single attribute of a single feature. This provides a product designer with a tool for performing precision redesign of a part.

The issue of setting a value for b merits discussion. This value must be set somewhere between 0 and 1, exclusive, such that p_4 has sufficient spread. One does not want to waste a good portion of the range of the function by setting b either too high or too low. Figure 26 illustrates the spread of p_4 for a range of b values. From Figure 12a one can see that the spread decreases as b decreases, therefore a higher value for b is desirable. Figure 27 shows the spread of p_4 for b values of 0.7 to 0.95, with q ranging from 0 to 50. Note that values for p_4 are calculated to the nearest one-thousandth for both Figures 26 and 27, and are shown to be zero if less than 0.001, even though this function tends towards zero at infinity.

Consider a part that has 100 form-features, all with moderate levels of difficulty associated with each feature attribute. Clearly this is a complex part by virtue of the great number of features involved. The accumulated penalty, q, may be in the neighborhood of 50, if all features are half-way up the difficulty scale. This is a very difficult part, but not necessarily infeasible, because each single feature is not really difficult. This illustrates the fact that part geometry, individual feature-types and orientations, and the number of features involved all contribute to part complexity. By accumulating penalties, one captures the complexity associated with increasing numbers of features. Since simple features such as cylindrical holes and cuboid through-steps are not penalized, unless toleranced or oriented other than 1D, one may consider setting b in terms of a number of moderately difficult features. The value of b is set at 0.95 for this development work, because a value of 0.95 provides extended spread to include over 100 moderately difficult features on a single part, while returning a decisively "non-zero" value to three decimal places and indicating severe inherent part design difficulty. Such a part design would not likely be selected for fabrication if an alternate design were available, or if redesign were possible. Please note that if the part design had specified only 50 features, but all with maximum difficulty ratings such that a feature

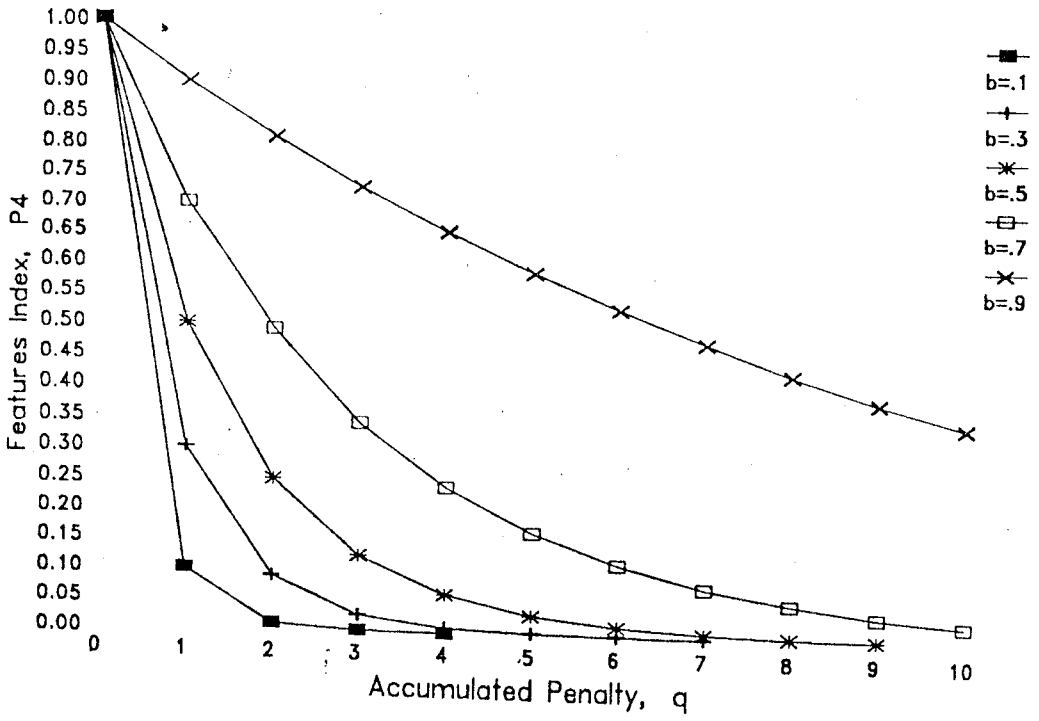


Figure 26: Features Index Versus Accumulated Penalty for various values of *b*.

penalty of 1.0 were assigned in each case, then the features index would be the same. Here one can see that the equation penalizes difficult features much more severely, as one would expect.

Also consider the case where all of the 100 hypothetical features are easy to fabricate, such as 1D-oriented cylindrical thru-holes and cuboid thru-steps with liberal tolerance and surface finish specifications. This part design would pick up very little penalty, and its features index would approach 1.0, since none of the difficulty parameters would register significantly. Recall that the producibility evaluation framework included cost estimation as part of the comprehensive evaluation. The total machining time for the part design would be used to help estimate the machining cost, which would increase with the machining time. Thus the cost estimate would reflect the demand on manufacturing resources that is made by even the easiest features.

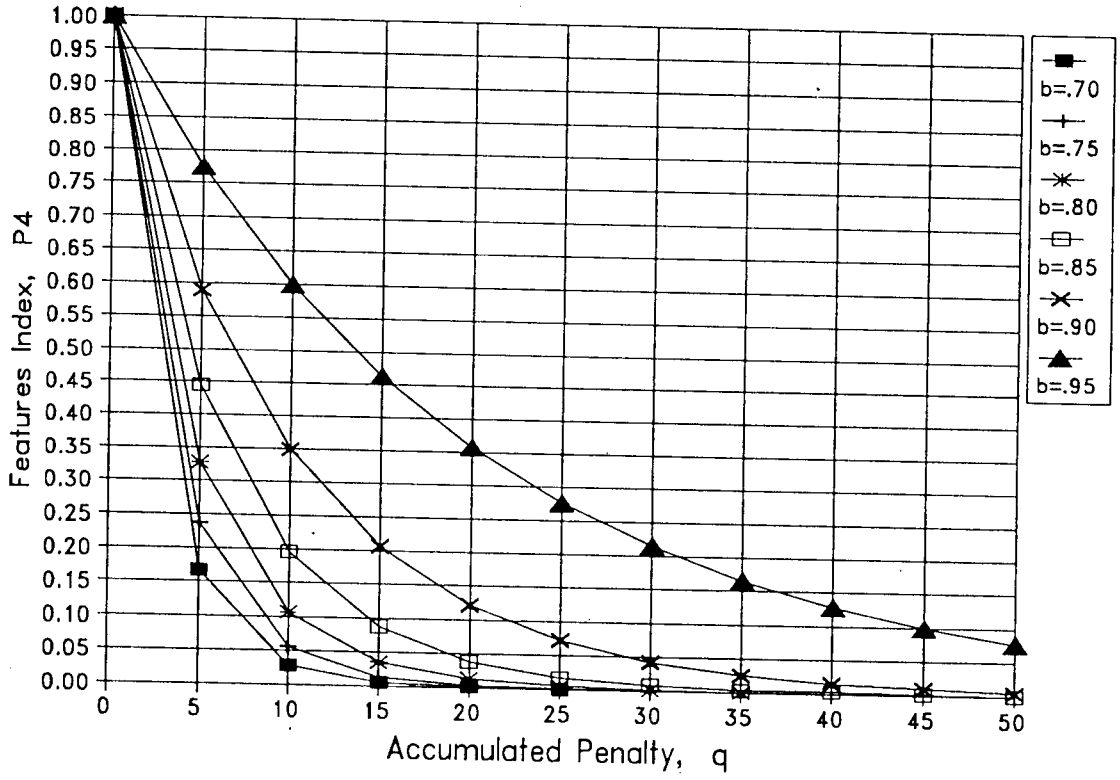


Figure 27: Features Index Versus Accumulated Penalty For Higher Values of *b*.

D3.4.1 The Base Part(0th Feature)

The base part is referred to as the 0th feature, because it is the first feature in the product model's feature list and is numbered zero. The special case of the base part considers only tolerance, surface finish and shape. The tolerance and surface finish specifications of the base part are used directly, but the shape of the stock material is used in place of the base part shape. The base part is currently cuboid for all cases, but the stock can be of several different shapes. The fact that the base part is always cuboid means that no information, with regard to relative difficulty between part designs, is available if the base part shape is used. Fixturing difficulty can be captured if the stock shape is used, because of the challenges presented by various stock geometries. If the base part shape were variable, then one may wish to use both the base part and the stock shape. This would capture the transformation of the stock shape into the part shape and the possible changes in fixturing requirements. This development is not in any way limited to part designs with cuboid base-parts. Stock shapes are defined as rectangular & octagonal, hexagonal, cylindrical and sheet (rectangular stock less than 1/4 inch thick). These categories are derived from a consideration of stock material shapes that are typically available, and are listed in increasing order of difficulty with regard to fixturing and handling on a machine tool. Sheets are considered to be most difficult because of their inherent lack of rigidity. Cylindrical Stock has limited stability, or requires surface modifications, unless fixtured in a V-block or three-jawed chuck. Hexagonal stock is likely to present edges to fixturing elements, while rectangular and octagonal stock are not. This makes hexagonal stock more difficult to fixture than rectangular or hexagonal.

D3.4.2 Individual Form Features

The measured difficulties for the individual form features are based on tolerance & surface finish, feature type and feature perpendicularity. The tightest tolerance(inches) and surface finish (μ-inches) specification is selected for each form feature evaluation. The tolerance and surface specifications are strong indicators of difficulty, because a limited number of machine and tool classes can accommodate tight specifications. Only the tightest specifications are used because they represent the worst case for each form feature.

The type of feature is derived from the primitive type used to make the feature, from whether the feature is a protrusion or a depression, and from whether or not the feature has sharp corners. There are currently over 100 named form features, each of which has been assigned to a group based on comparable difficulty. Certain feature types are more difficult to fabricate than others. The primitive shape, feature topology (depression/ protrusion), and corner radius condition are the primary factors that influence which group that a given feature is assigned to. For example, a prismatic, sharp cornered pocket would be considered to be more difficult than a rectangular boss. Blind depressions with sharp internal corners are among the most difficult, and must be differentiated from the easier features. Through-depressions and protrusions are among the easiest to fabricate, unless the primitive type is difficult to fabricate, such as an elliptical thru-step. An elliptical thru-step is a step on a part that has an elliptical surface

profile.

The perpendicularity of a feature is established according to the rules used in the feasibility assessment, and in addition, 1D-orientations are assigned no difficulty and 2D-orientations are assigned less difficulty than 3D-orientations.

Equations 7 and 8 capture the difficulty of a given feature as follows:

$$f_j = \frac{1}{C_j} \sum_{k=1}^{M} c_{j,k} d_{j,k} \tag{7}$$

$$C_j = \sum_{k=1}^{M} c_{j,k} \tag{8}$$

Where C_j is the sum of weights of importance for each feature attribute, $c_{j,k}$ is the weight of importance for the kth attribute of the jth feature, $d_{j,k}$ is the difficulty assigned to the kth attribute of jth Feature and M is the number of attributes. For the base-part M = 3 and for the other features M = 4. Recall that only tolerance, surface finish and feature-type are considered for the base-part.

D3.4.3 Clearance Between Features

Clearance between features is computed by the same method as for technical feasibility assessment, but there are no special considerations involved. The feature pair with the minimum non-zero clearance is identified and this minimum clearance is used for applying the penalty. The difficulty associated with clearance is based on the challenge of negotiating a cutter tool between two features that are very closely situated. The clearance of the tool within this space is the basis of this criteria. A tool must be able to fit within the minimum clearance, otherwise it is blocked. Thin-wall conditions are also captured by this parameter. The minimum distance between two depression features can be a thin-wall condition. Thin-walls can cause reduced product yield due to material failure in the wall, and can also imply difficulty for EDM processes, because the thin-wall translates into a narrow depression feature or gap on the EDM electrode. This is because the electrode is analogous to a negative of the form feature that it is used to fabricate. This is one reason why EDM is considered to be more difficult than traditional machining. The electrode must be machined before the part can be processed, which involves producibility evaluation of the electrode, as well as subsequent process plan generation and NC programming. The minimum clearance between form feature pairs is used for this penalty for either of the two cases; thin-walls or tool path negotiation. The penalty for clearance is assigned to $f_{(j=N)}$, which is the last penalty accumulated into q.

D3.5 Procedure for Computing the Producibility Index

Now that the parameters have been introduced and discussed, and the equations for computing the producibility index are developed, please refer to Figure 28 for an illustration of the procedure for computing the index. The first two steps are to link to

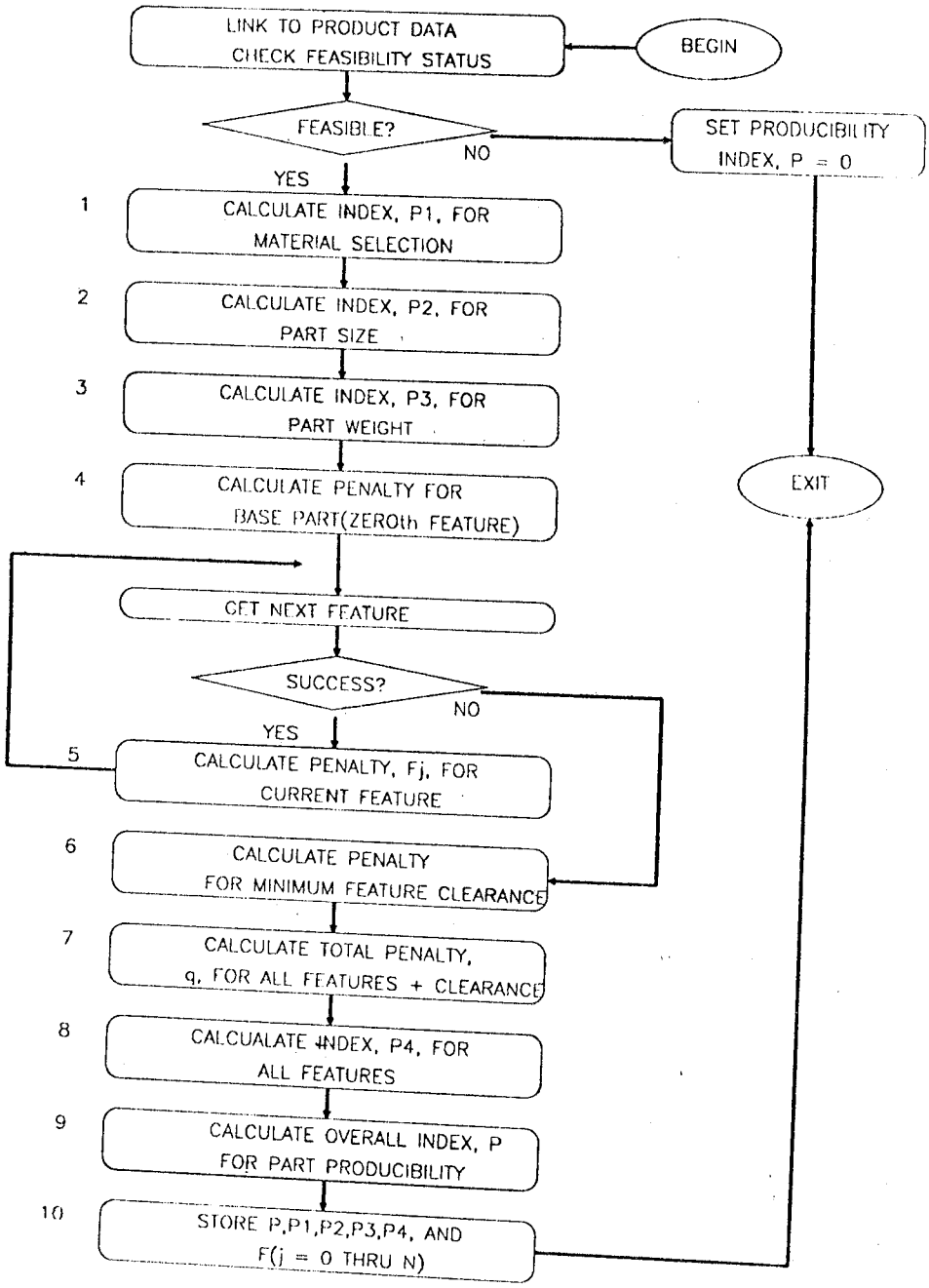


Figure 28: Procedure for Computing the Producibility Index.

the product data, and to check the feasibility status. If the status of the part design is "feasible", then the procedure continues, else it terminates. Activity number one is the computation of the material selection index, p_1, per equation 2. The part size and part weight indices, p_2 and p_3, follow in succession and are computed according to equations 3 and 4, respectively. Activity number 4 is the computation of the penalty for the base-part, which is made according to equations 7 and 8. For the case of the base-part the value of M is equal to 3, which allows for consideration of the minimum tolerance specification, smoothest surface finish specification and feature-type parameters, from which a weighted mean penalty is based. After the base-part penalty is computed, the procedure enters a loop that computes a penalty for each subsequent form feature according to equations 7 and 8. Recall that M is equal to 4 for all features except the base-part, because of the added orientation parameter. Thus, activity number 5 is repeated for N-1 features. Because the base-part is counted as feature number j = 0, the last feature is numbered j = N-1 and the minimum feature clearance is numbered j = N. The penalty assignment for clearance is therefore the last penalty assigned. The total feature penalty is next computed according to equation 6, and then the features index, p_4, is calculated according to equation 5. Now that activity number 8 has been completed, the composite index can be computed according to equation 1. The producibility indices and the individual penalties, f_0 through f_N, are then stored for later use. The producibility evaluation data is stored with the part ID, part version and the evaluation date.

D4. Summary of the Development

The method development began by presenting a producibility evaluation framework that was then examined for elements that could be developed into fully detailed and functional methodologies for implementation. Technical feasibility assessment, design producibility rating, cost estimate, quality rating and schedule compatibility assessment are the elements of the comprehensive producibility evaluation framework. The technical feasibility assessment and design producibility rating elements, enacted prior to process plan generation, are the chosen elements for development. For clarity, a class structure is presented in Figure 29 that illustrates the framework and details the developed elements. The icon labeled "Producibility Evaluation" represents the superclass of the structure. For example, a technical feasibility assessment is a producibility evaluation, a design rating is a producibility evaluation, a cost estimate is a producibility evaluation, etc. All producibility evaluations inherit the part ID, part name and part version from the product model as a necessary means of properly linking the evaluation to the correct product design. Each evaluation could be made on a different date, so a separate date attribute is required for each class of producibility evaluation.

The technical feasibility check has 1 global check and 1 or many feature checks. The checks provide information in the form of the feasibility status and additional comments. A feasibility check for a feature has the added attribute of the feature ID so that the status of a given feature can be effectively linked to the feature. The status of the part design is determined by performing a systematic search for feasible material,

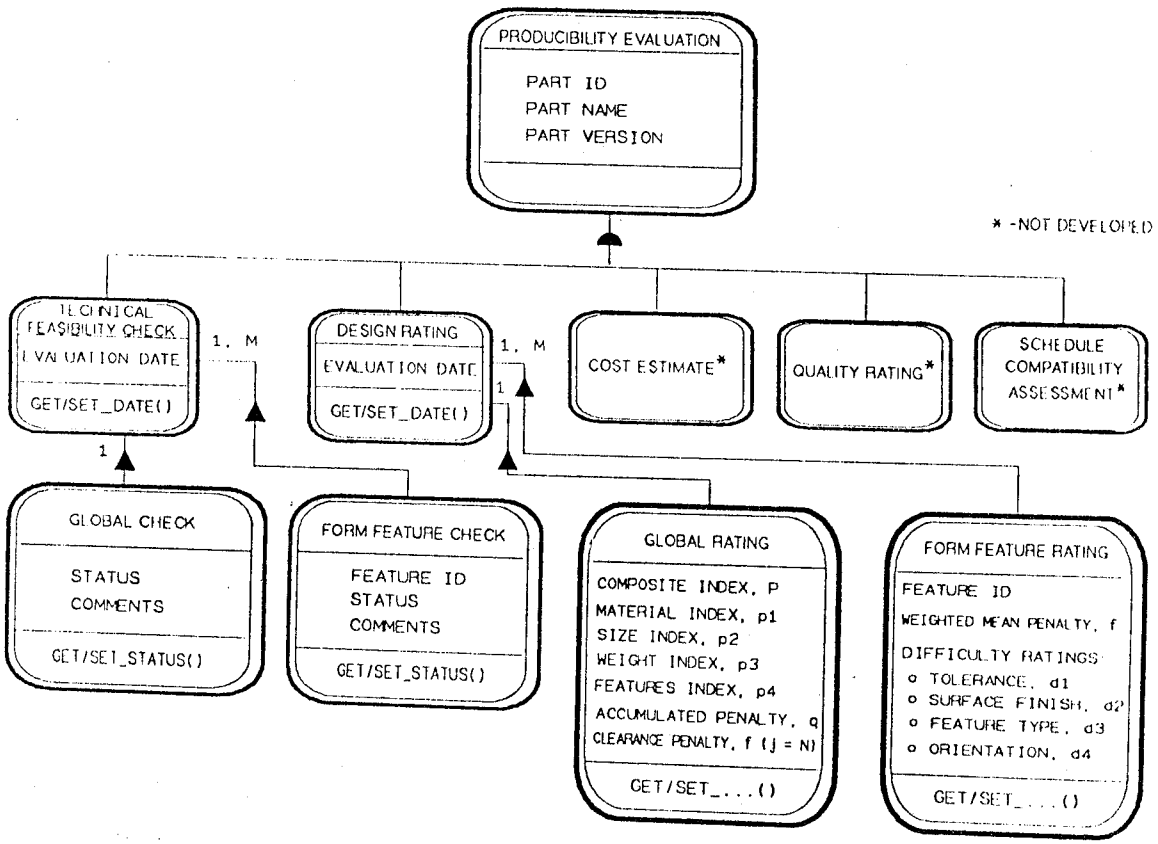


Figure 29: Producibility Evaluation Class Structure.

machines and tools for fabricating every form feature per plans and specifications.

The design rating is the producibility index developed in this chapter, and it is also a producibility evaluation. The design rating has 1 global rating and 1 or more feature ratings. The global rating has attributes the include the composite producibility index, P, the material selection index, p_1, the part size index, p_2, the part weight index, p_3, the features index, p_4, the clearance penalty, f_N, and the accumulated penalty, q. Each feature rating has a feature ID, a mean weighted penalty and four difficulty ratings; d_1 for tolerance, d_2 for surface finish, d_3 for feature-type, and d_4 for orientation. The base-part is an exception to this general structure, because the orientation parameter is not addressed. Such a structure as the producibility framework, particularly the developed elements, may be readily implemented in a feature-based object-oriented environment, as is the case.

D5. Implementation and Validation

This method could be implemented manually with step by step instructions regarding when and where to reference tables and charts, and which equations to use and in what order. This would be very time consuming for even a simple part with but a few form features, especially when performing the feasibility assessment. An improvement over manual implementation is interactive computer implementation. A computerized version of the manual could prompt the user for information about the part model, and would progress according to the user's response. This is quicker than the manual method, but would still be time consuming and tedious because of the delay between prompts and responses. A fully automated computer implementation is the most efficient way to utilize this method. Automation saves time and saving time saves money, which is the most typical justification for new technology such as this. This method has been implemented as a fully automated computer system to make it practical.

The method has been implemented in C++. This software is supported by a network of work stations, dominated by a Sun/SPARC10, which is host to the Unix-based Integrated Product/Process Development System Software located in the Concurrent Engineering Research Laboratory(CERL) at Florida International University. This system is composed of a custom CSG/B-rep product modeler(for details refer to Chen and Wu [6]), a feature extractor, a tolerance modeler, manufacturing resource manager and knowledge manager. An object oriented database is used for storing product data, manufacturing resource data and rules for the knowledge base. This system represents the software and is illustrated in Figure 30. The applications software programs are shown on the right side of Figure 30, while the support software programs are shown on the left. The various classes of data that are stored in the object-oriented database are shown in the center of Figure 30. For a detailed description of this system refer to Chen, et al. [7].

One critical element of the system, with regard to supporting producibility evaluation, is the manufacturing resource database. This database contains material, tool and machine instances of their respective classes, with their respective attributes. Material attributes include dimensions, shape, material type, AISI code, availability

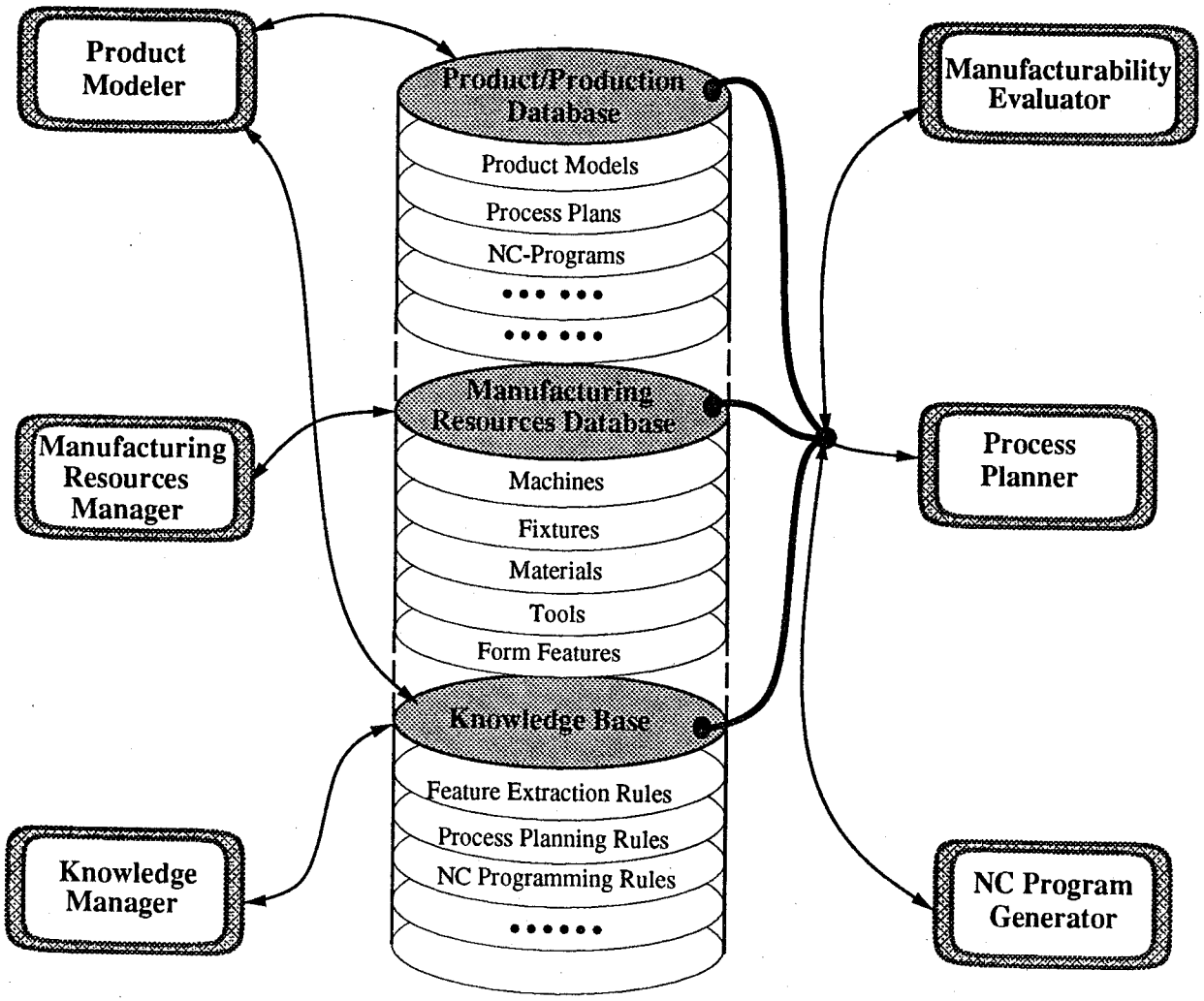


Figure 30: Integrated CAD/CAM System Configuration

status, and source. Weight per foot, cross-sectional area, and the total weight of the stock piece are available, as well. The tool attributes include tool type, tool holder type, maximum tool assembly dimensions, material cut capability, surface finish capability, cutter dimensions, insert information, and other information. The machine attributes include machine type, work envelope dimensions, table weight capacity, tool travel capability, tool holder type, position accuracy & repeatability, and other information. This information supports the procedure for identifying feasible stock material and machine/tool combinations.

Another critical element is the computer implemented feature-machine-tool(FMT) table, which permits rapid identification of feasible FMT combinations at the class level. This capability greatly simplifies the search algorithms, by acting as a buffer to prevent undesirable combinations from being considered. The FMT table is implemented as a list of form-feature classes, each with a list of compatible machine tool classes. Each machine tool class has its own list of compatible tool assembly classes. This structure permits cross-referencing at the class level. The FMT table is located in the knowledge base of the object-oriented database.

The product modeler is the means by which product data is supplied to the evaluation program. Product data is saved to the object oriented database, where the producibility evaluator will have access to it. Product data includes the part ID, part name, version number, material type, AISI code, designer's name, the design date, and any tolerance reference datums. In addition to the part data, there is a set data for each form feature. The feature data includes feature ID, feature type, feature orientation, feature rotation, feature dimensions and location, tolerances and surface finish specifications, the parent feature, and a list of child features. There is also information about the faces of each feature, such as their orientations and whether they are trimmed faces or not. A trimmed face is a face such as the opening to a cuboid pocket, while the remaining faces represent the walls of the feature.

The feasibility check is made by an on-line command, *feacheck*, that is made from a regular X-window on the screen, and the producibility index is called from a custom X-window. Figure 31 illustrates the window for computing the index. The system is fully automatic. The user is first presented the option of selecting a product model from the database. Upon making the selection of a part model, the user is prompted to choose one of three options: (1) display the producibility already stored in the database; (2) calculate the producibility for the part model; or (3) save the producibility to the database. All other operations are done automatically by the program.

Even though the implementation of the technical feasibility method followed the intent of the flow-charts very closely, there are some points that need to be explained. The implemented version tests only 2 1/2 D form-features that have 1D-orientation. If the feature-type is not recognized by the system, then the feature is flagged as infeasible and the part design fails the feasibility test. A message is sent to the X-window that an unknown FMT feature-type has been identified. If the feature orientation is not 1D, then the program flags the feature as infeasible and sends a message that the feature is not 1D- oriented. If the material specification does not match any listed in the material database, then the design fails the test, and a message is sent informing the user that

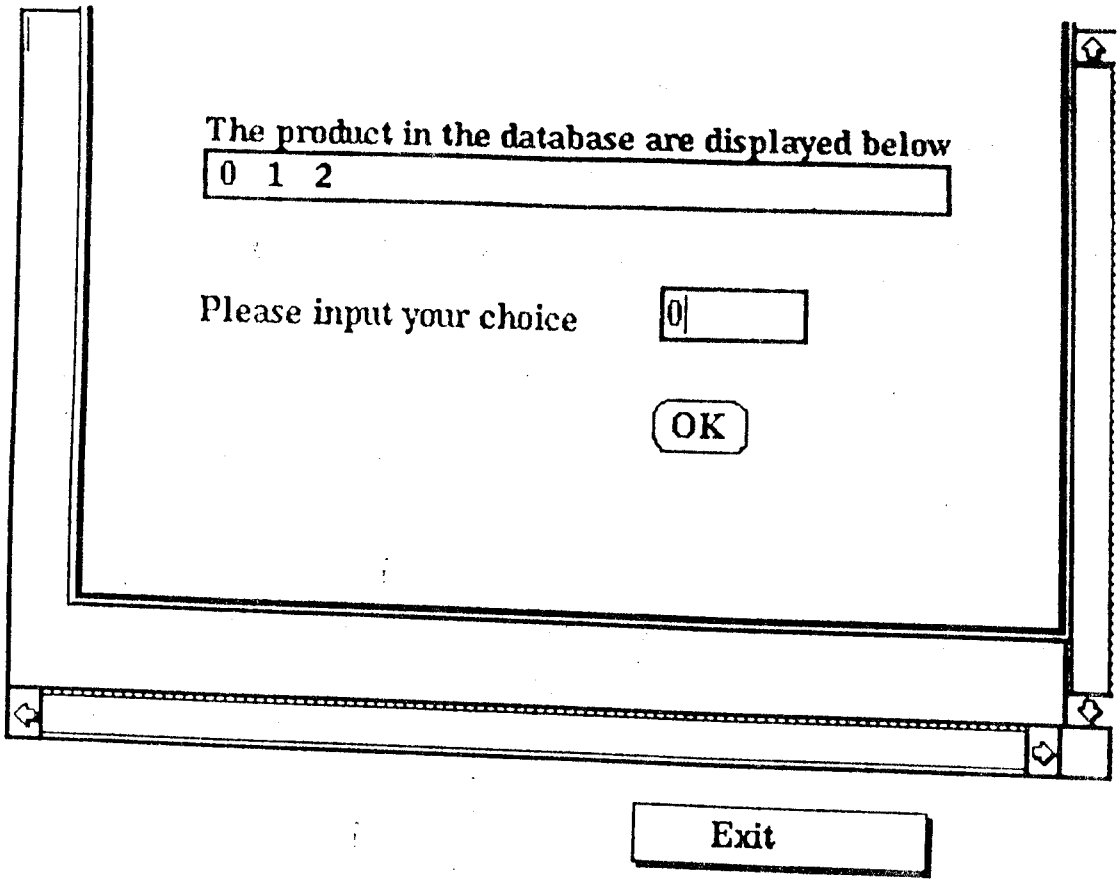


Figure 31: User Interface for Producibility Evaluation.

either the material type or AISI code is not found, whichever is the case.

D5.1 Parameter Value Assignments

In this section values are assigned to the parameters established in the producibility evaluation model. A brief explanation will accompany each assignment. The weights of contribution are established according to relative rank, where a higher ranked parameter is assigned more weight, because the more important parameters are assumed to have more impact on producibility and the weights reflect this. The range of difficulty evaluations for feature parameters is set from 0 to 1 in order to remain consistent with the mechanics of the producibility equations, which also have a range of 0 to 1.

D5.1.1 Global Parameter Value Assignments

Threshold size is set at 3 inches (United States Customary Units(USCU)). Recall that size is defined as the sum of the part's length, width and height. A cube one inch on a side has a size of 3 inches, which is a relatively small part. A size of 3 inches reflects the author's opinion regarding threshold difficulty, and is based on the perceived handling and setup difficulty that such a small part represents. This size part would be small compared to most fixturing elements that may be used to constrain it. Persons with experience in the machining process may very well have their own opinion regarding the value of this parameter.

The threshold weight is set at W_t = 50 pounds USCU, and represents the upper limit for pick and place transfer by manual means. This figure is in agreement with current ergonomic standards reported by Waters, et al. [8]. The maximum weight, W_{Max}, is set equal to highest machine table weight capacity in the database, presently 1000 pounds, because no feasible part can weigh more than this amount. Finally, the actual part weight, W_a, is estimated by the product of the volume and density of workpiece, where the rectangular volume of the entire part envelope is used. This is a conservative assumption, because the actual weight of the part decreases with material removal, however, part weight is a global parameter so the greatest weight of the part is used.

D5.1.2 Feature Parameter Assignments

The base part has its own set of parameter values. Tolerance and surface finish difficulties are the same as for the other features, but the weighing factors are different. The base-part is penalized based on the stock shape. Sheet is penalized with the maximum value of 1.0, cylindrical is penalized 0.7, hexagonal is penalized 0.2, and rectangular and octagonal are penalized 0.0. Sheet stock is penalized the most because it is flexible, cylindrical stock is strongly penalized because its roundness makes it more difficult to restrain except by a three-jaw chuck or V-block, and hexagonal stock receives a minor penalty because it has the potential for presenting an edge, rather than a plane surface, to a fixturing element. The weights of contribution: for tolerance $c_{0,1}$ = 2; for surface finish $c_{0,2}$ = 2; and for part shape $c_{0,3}$ = 1.

The penalties for tolerance and for surface finish are provided in tables III and IV. Weighing factors for feature evaluation are: for tolerance $c_{j,1}$ = 4; and for surface finish

$c_{j,2} = 4$. The values for tolerance are based on TMEH [9] and the values for surface finish are based on TMEH [10]. Note that the maximum penalty for tolerance is assigned to what is commonly considered a high precision tolerance. The maximum penalty for surface finish corresponds to a mirror-like surface.

Table III: Difficulty Value Assignments for Tolerance.

tolerance	≥0.01	≥0.005	≥0.001	≥0.0005	<0.0005
difficulty, $d_{j,1}$	0	0.25	0.50	0.75	1.00

Table IV: Difficulty Assignments for Surface Finish.

finish	≥128	≥63	≥32	≥16	<16
difficulty, $d_{j,2}$	0	0.25	0.50	0.75	1.00

Difficulty for feature type is assigned according to table V. The weighing factor is: $c_{j,3} = 2$. As new feature types become available they will be placed in the appropriate group according to the criteria discussed in section D3. If an undefined feature is encountered then a default penalty corresponding to that of group 5 is applied, as a conservative measure.

Table V: Difficulty Assignments for Form-feature Type.

group	1	2	3	4	5
difficulty, $d_{j,3}$	0	0.25	0.50	0.75	1.00

Difficulties for perpendicularity are: 1D-orientation→$d_{j,4}$ = 0; 2D & 3D-orientation→$d_{j,4} = 0.7$. Currently, the program only differentiates between 1D-orientations and those that are not. Since there is no differentiation between a 2D-orientation and a 3D-orientation, a value of 0.7 is used so that the net penalty for all feature orientations is neither under evaluated nor over evaluated. The weighing factor is $c_{j,4} = 1$.

Clearance of 1/8 inch or less implies either very small traditional machining tools, or EDM machining. Clearance of less than 1/16 inch implies possible difficulty in fabricating an EDM electrode, because a 1/16 inch gap on the part correlates with a 1/16 inch thin-walled region on an EDM tool. For clearance greater than 1/8 inch the penalty is 0, for clearance between 1/16 and 1/8 inch the penalty is 0.5, and for clearance less than 1/16 inch the penalty is 1. Here, f_j for clearance is the Nth f-value computed.

D5.2. System Evaluation

In order to determine if the computer programs were performing as intended, some testing was done. This testing also provided a basis for judging the performance

of the method. A brief description of the testing will begin with the technical feasibility program and end with the producibility evaluator. In most cases the parts used for testing the feasibility and producibility programs were the same, though some additional parts were tested by the feasibility program to check tool clearance and travel. Verification of the technical feasibility tests was made by either setting up a part design that was known to be infeasible for a specific reason, such as including a square-cornered pocket in the design, and/or by reviewing the stock material, machine and tool instances that were selected for each feature. If the selections made sense, the test was verified. Remember that there may be many feasible machine and tool combinations for the same form-feature. Verification of the producibility program was made by calculating the expected results and then comparing them with those obtained by the program. Since a Sun workstation computes to more significant figures than a hand calculator, the two sets of values were not expected to be identical, but did have to be close. For example, a calculator value of 0.88756 for the features index was considered verification of a corresponding computer value of 0.88568.

The initial testing of the feasibility program involved checking if the program recognized different features by evaluating several simple part designs on the system. These simple part designs consisted of a cuboid base part with one to three additional form-features. Once the system's feature capability checked out, the next step was to test the ability to execute the higher level functions, such as tool clearance checking. The example parts provided in this section were used to test the feasibility program with regard to its ability to check tool clearance, tool travel, work envelope capacity, tolerance and surface finish capability, corner radius capability, and feature depressions against tool dimensions. The examples will be discussed later in this section.

The producibility evaluation program was tested with part designs that were used to test the feasibility program. Size and weight parameters were checked by varying the size of the base part, and by using materials of different density. Machinability was tested by using different material types for the same part geometry. The extent of the testing is illustrated by a few selected example part designs that will be discussed next.

Part0 of Figure 32 is a simple cuboid block with surface finish and flatness specifications. This block demonstrates the capability of the system to capture tolerance information, and with the proper tolerances, could have real-life application as an optical component. Mirrors and reflectors are commonly machined from aluminum and require unique surface specifications. Part0 has a minimum surface roughness of 25 μ-inch, and a minimum flatness tolerance of 0.001 inch. The feasibility program selected rectangular stock, and identified the availability of a mill type machine and endmill. The producibility index is 0.891, which is considered highly producible.

Part1 of Figure 33 is used to test the clearance checking segment of the feasibility program. Two rectangular bosses are provided a relatively wide clearance. Rectangular stock is selected, and a mill type machine and endmill are selected. The producibility index is 0.900. Though this part requires the milling of two additional form features, it is as easy to fabricate as Part0, because tolerances are not specified and are therefore not penalized.

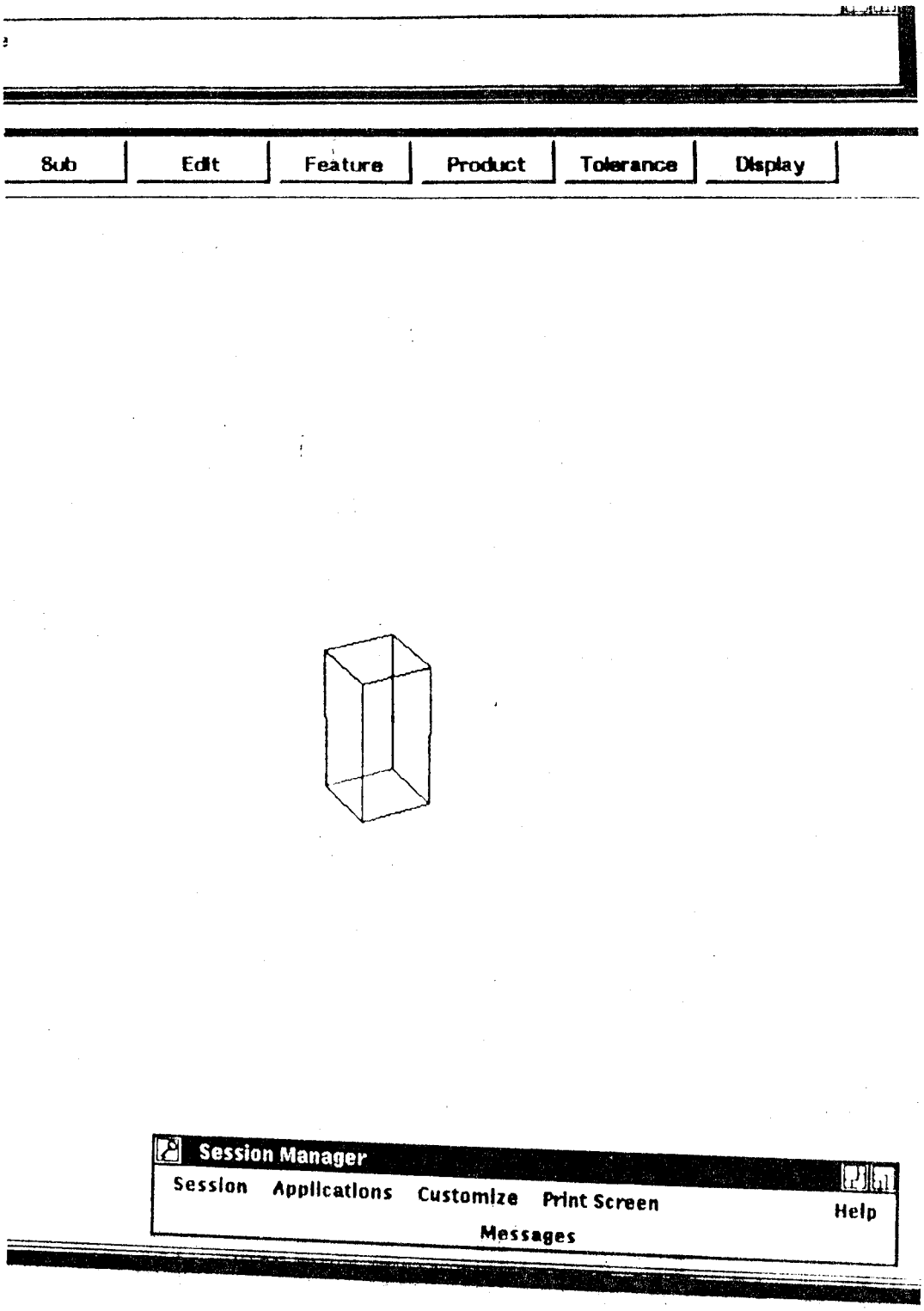


Figure 32: Part 0.

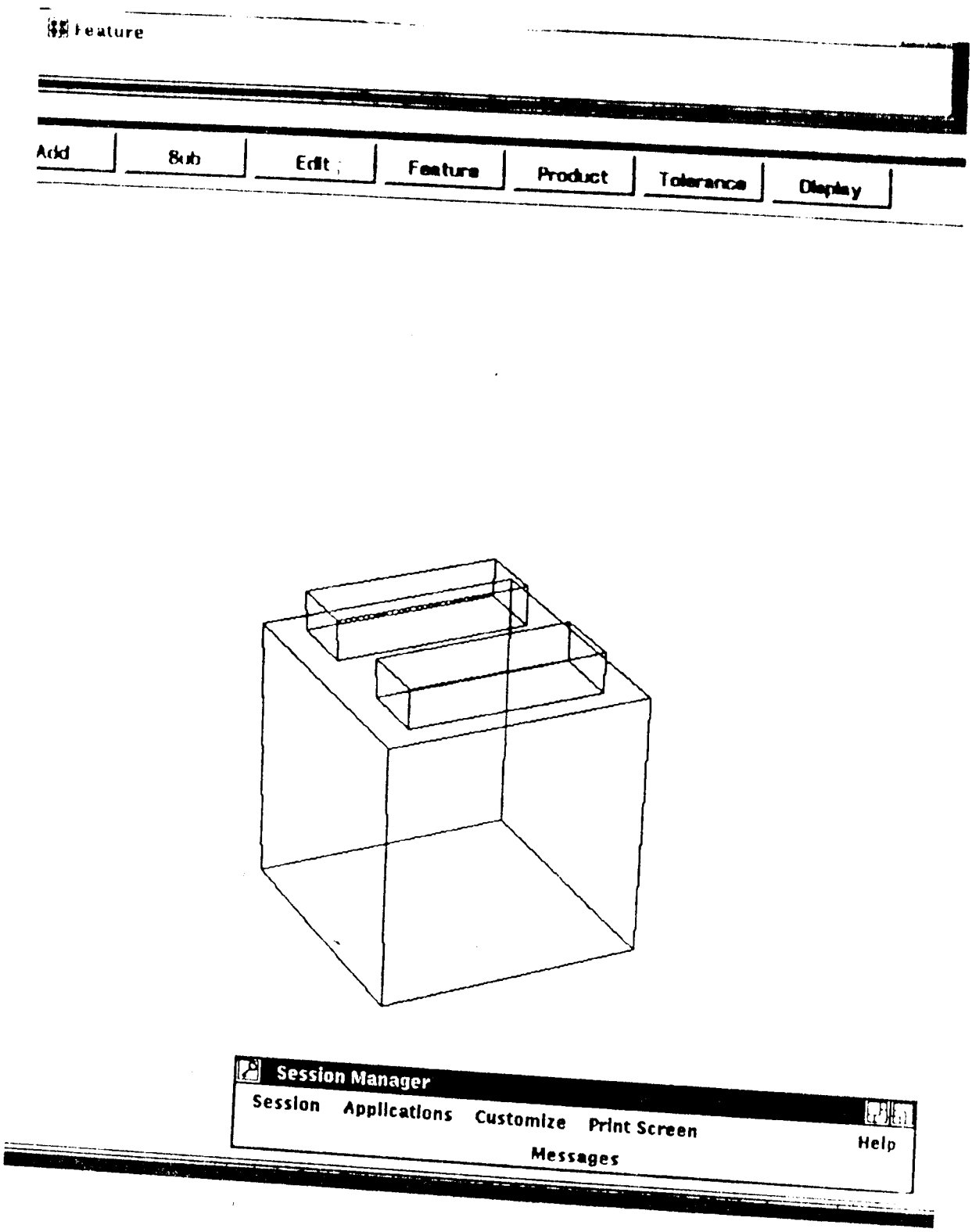


Figure 33: Part 1.

Part2 of Figure 34 and Part3 of Figure 35 are used to test for differentiation between sharp-cornered and radius-cornered depressions. Clearance between form features is also tested. Part2 and Part3 are virtually identical, except that the blind cuboid depressions are square-cornered and radius, respectively. Consequently, Part2 is not feasible, while Part3 is feasible. Part 2 is infeasible because no feasible machine/tool combination exists in the database. EDM can produce these features, but there is no EDM electrode with suitable dimensions. This is one of those cases where either the part must be modified, or the manufacturing resources enhanced. If one considers Part3 to be a modification of Part2, then the concept of precision part modification has been demonstrated. The producibility index for Part3 is 0.889. Note that tolerance was not considered here, which contributes to a very high producibility.

Based on the results of the tests, the overall capability of the system is good. The system has some trouble dealing with features that are not oriented along a coordinate axis, because of a peculiarity of the product modeler. The modeler places each feature within a cuboid bounding box, and the vertices of the bounding boxes of any two features are used to measure the minimum distances between them. If the feature is not cuboid there is automatically a possibility for discrepancy between the true minimum distance and the one obtained via bounding box vertices. The same discrepancy occurs, even for cuboid features in this case, when two features are on a wedged face and are at different locations along the slope. In this case the bounding boxes of each feature maintain an orientation along a coordinate axis, while the features are oriented along the normal to the wedged face. In order to place a bounding box around a form-feature, the points needed for computing the true minimum distance must be known somewhere in the program, but are not available for use by applications programs. The true minimum distance is always greater if there actually is a difference. See Figure 36 for illustration.

The system can not take full advantage of EDM capability, because most EDM tools are custom items and are not supported by the manufacturing resources model. Only cuboid and cylindrical shaped features can be considered for fabrication by EDM electrodes. The system is further limited to using existing tool assemblies, because neither the feasibility program nor the manufacturing resource model presently handle tool assembly synthesis. Future expansion of the producibility evaluation methodology and implementation may require a method of calling up custom EDM and traditional tool assemblies.

The producibility program returns values nearly instantly, but the feasibility assessment program can take a few minutes to evaluate a part with many features. The assessment time duration is a result of the many nested loops that are used for checking all of the feasibility parameters. For the purposes of testing, the number of tool, material and machine instances in the database was kept relatively low. The execution time of the feasibility program can be expected to increase as more objects are loaded into the database, because the loops will become longer.

D5.3 A Walk-Through Example

Part4 of Figure 37 has three versions that are all geometrically identical. The

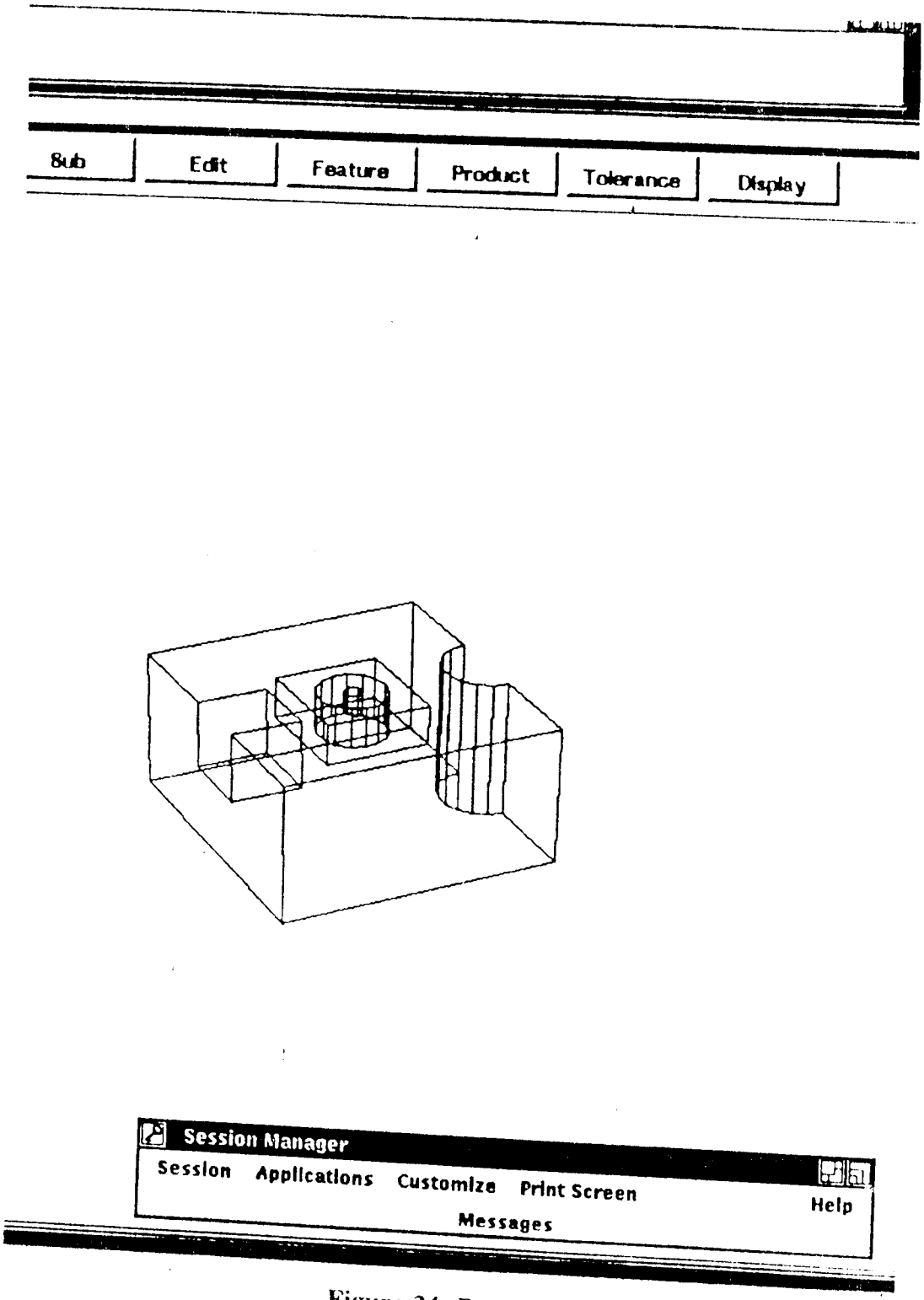


Figure 34: Part 2.

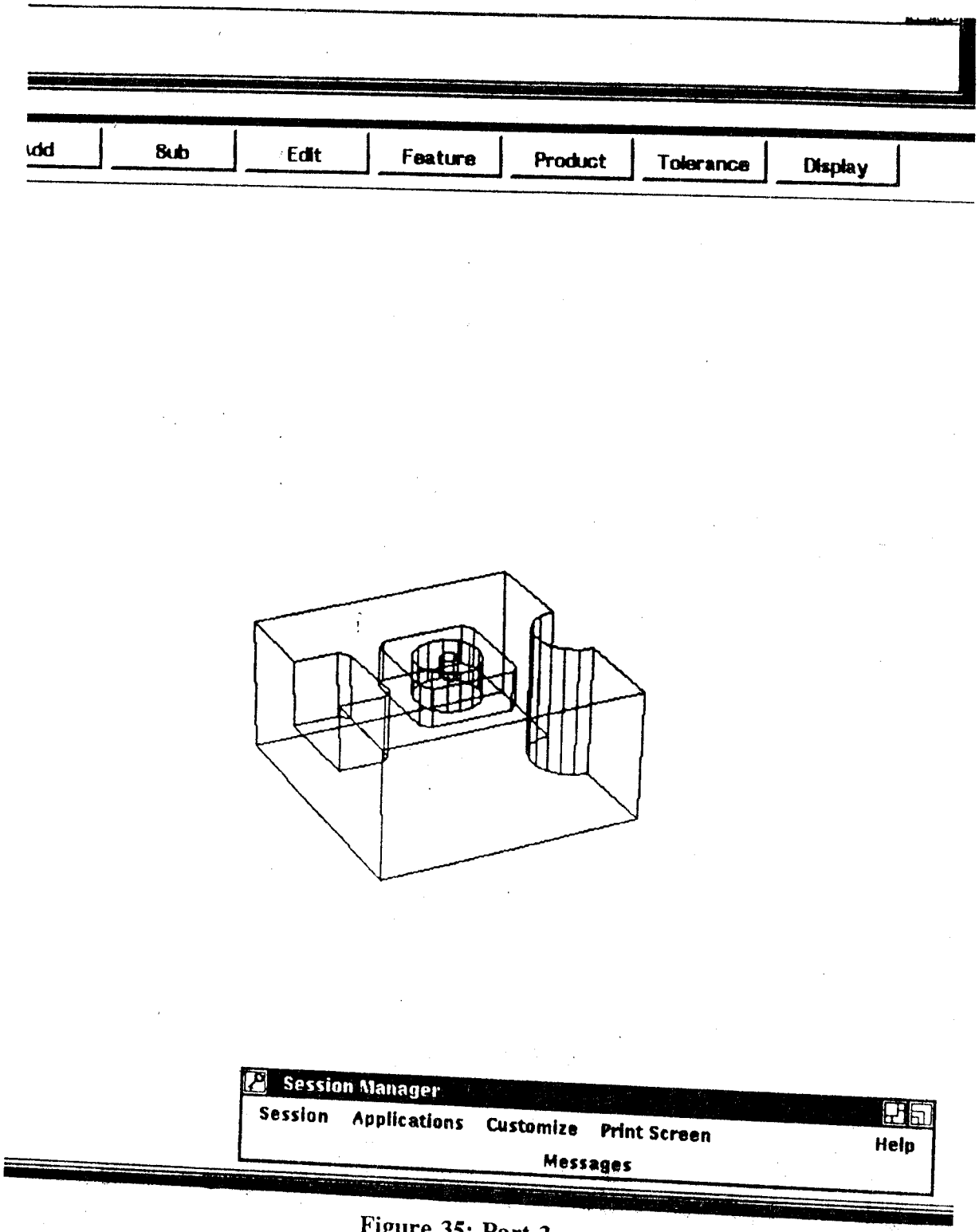


Figure 35: Part 3.

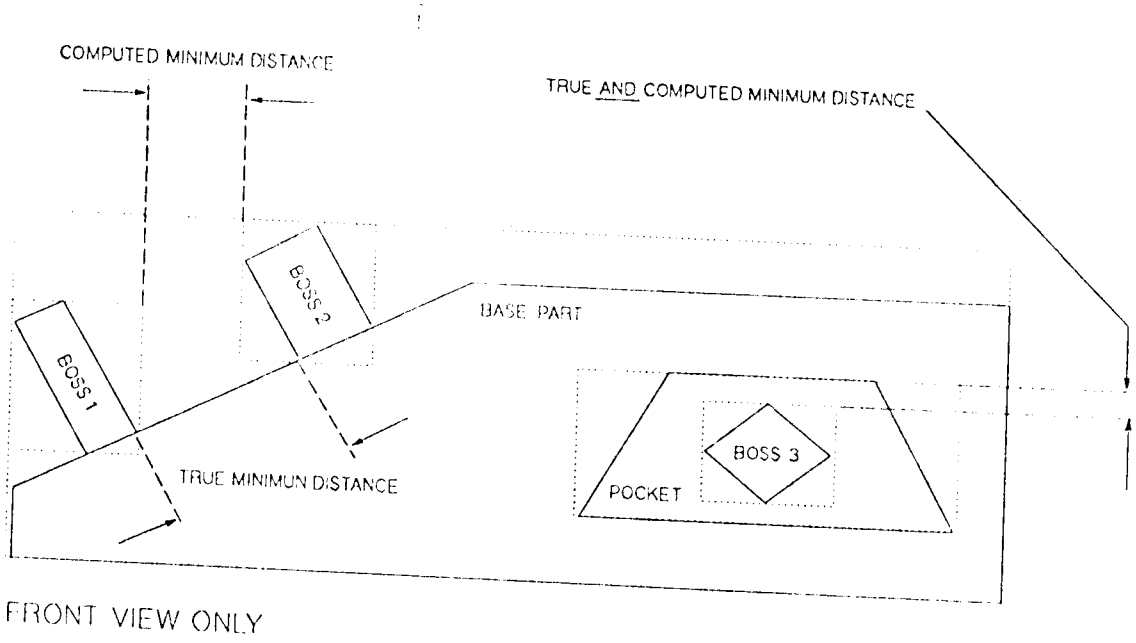


Figure 36: Illustration of Error in Tool Clearance Check.

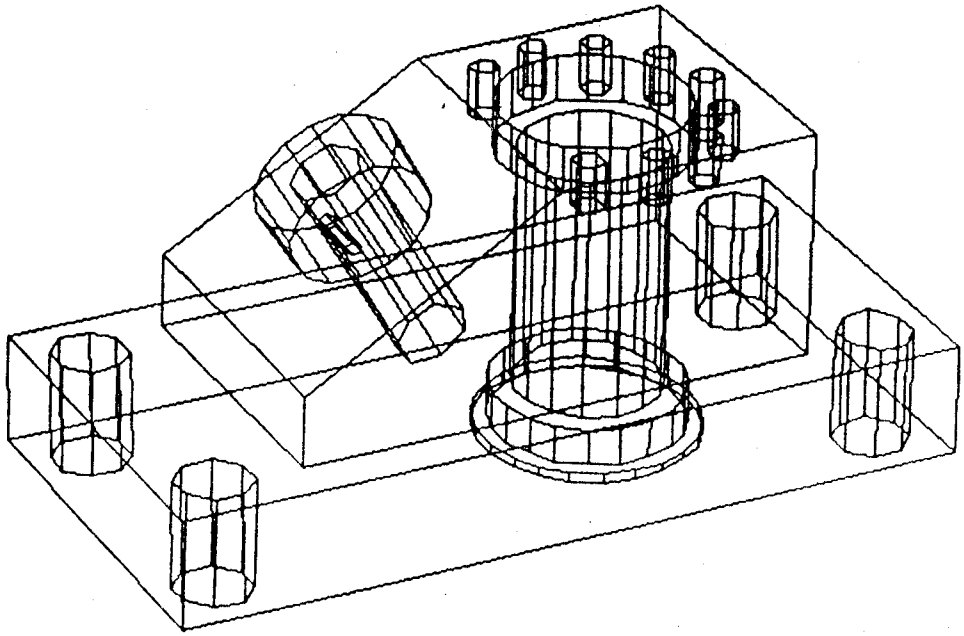


Figure 37: Part 4.

differences between the three versions are distinguishable by: (1) Aluminum 5005 with no tolerance specifications; (2) Stainless Steel 414 with no tolerance specifications; (3) Stainless Steel 414 with a position tolerance of 0.01 inch assigned to the large cuboid boss, and a minimum surface finish of 35 μ-inch specified for the wedged through-step located on the large boss.

For version 4.1, rectangular aluminum stock was chosen by the feasibility program, and a rectangular stainless steel stock was chosen for version 4.2 and 4.3. The feasibility program chose a milling machine and an endmill for the part level assessment, as well as for each of the form feature assessments. All of the features on Part4 are rather simple, and feasibility was not an issue. The goal here is to demonstrate the performance of the producibility index program.

Part4 shows how the producibility index responds to changes in material type and tolerances. The change in producibility from Part4.2 to Part4.3 is small, which was expected. Since the position tolerance does not pick up a penalty, there is only one tolerance affecting the entire part; surface finish. If the response had been large for only one tolerance specification the features index would decay to nearly zero for a modest set of tolerance specifications. As it is, the index can gradually accumulate many tolerances. For example, if the same surface finish specification were assigned to all 23 features on Part4.3, the resultant features index would be 0.909 and the total index would be 0.651. Part4.3 provides a good example of the amount of sensitivity that is available in the

Table VI: Producibility Values for Example Parts 0 Through 4

Part Name	Material Index, p1	Size Index, p2	Weight Index, p3	Feature Index, p4	Total Index, P
Part0	0.9	1	1	0.989794	0.890814
Part1	0.9	1	1	1	0.90
Part2	0	0	0	0	0
Part3	0.9	1	1	0.988040	0.889203
Part4.1	0.9	1	1	0.988004	0.889203
Part4.2	0.716667	1	1	0.988004	0.708069
Part4.3	0.716667	1	1	0.979101	0.701689

program. The producibility indices for the parts are provided in table VI.

Note from Table VI how the producibility indices for size and weight are unity

for all examples, this is because all of the example parts are well within the threshold limits that have been set for size and weight. Further note that a part will rarely be penalized for size or weight, and even more rarely will a part be penalized for size and weight simultaneously. Small size and heavy weight tend to be mutually exclusive states.

D6. Conclusion

This chapter is composed of two sections. The first section is a summary of what has been done and the results of the development and implementation, and the second section highlights the areas for future research.

D6.1 Summary

A method for producibility evaluation prior to process plan generation has been developed and implemented into a producibility evaluation system. The method evaluates prismatic parts with 1D-orientation features by checking technical feasibility and then computing a composite producibility index that is based on difficulty. Example parts that demonstrate the capability of the system have been provided in Chapter 4.

A candidate part is determined to be technically feasible if suitable stock material is found, and machine/tool combinations are identified such that every form feature on the part can be fabricated. Infeasible form features are identified and recorded. By demanding a direct match between design intent and manufacturing capability, this method is dynamic rather than static. A static method may employ a set of manufacturing rules that are used to determine feasibility, but this type of assessment is not adaptable, and requires continuous revision of the rules in order to stay current. The method presented in this thesis requires an inventory of stock materials, machines and tools for checking feasibility. A new item is merely added to the inventory, and it can then be used in the feasibility check. If the item is not within the scope of application, then it can not be added until the methodology and implementation program are expanded.

The method is comprehensive in that it checks feasibility and evaluates producibility of individual form-features, as well as global parameters associated with the entire part design. The ability to capture manufacturing difficulty at the attribute level of a single feature makes this a precision tool for product redesign. This precision tool can streamline product design by allowing the designer to focus on problem areas within the domain of a single feature. The producibility index is particularly powerful because of its simplicity, range of evaluation, and its ability to be customized to meet a particular user's manufacturing environment. The index can be customized by adjusting the parameters in the equations for P_1, P_2 and P_3. For example, in P_2 one may wish to use a different threshold size than 3 inches. The index can be further customized by setting the value of b in P_4. The value of b affects the spread (useful range of the function) of P_4 by making the equation either more or less sensitive to the cumulative feature penalty. Since this is an exponential decay function, a higher sensitivity results in quicker decay and smaller spread, while the opposite occurs for lower sensitivity. As b approaches 1.0 the spread increases. Finally, one may customize the index by

establishing proprietary values for the difficulty and weight values of each feature attribute. Customizing the producibility index should be done by an experienced person who understands the processes involved, just as a skilled musician tunes an instrument or a marksman sights in a rifle.

In order to avoid process planning activities, certain simplifications and generalizations were made. These simplifications are considered to be reasonable and valid. The result is a method that acts as a net or filter to capture major feasibility issues. The major simplifications are discussed next.

The feasibility assessment is oriented towards "in-house" facilities, which makes it a local system. This is the first major simplification. A generic assessment would need to consider all tools and machines available in the world. No single manufacturer owns every type of machine and tool known to mankind, so this is a practical simplification.

When searching for a feasible machine or tool, the program looks for those machines and tools that can make the finished feature. For example, a large mill can "hog out" significant volumes of metal from a rounded depression, but only a cutter with a radius less than or equal to the feature's corner radius can finish the feature. Process planning may consider staged milling, or pilot hole tools, but this feasibility assessment does not. This is the second major simplification.

Diversity in tool shapes represents tremendous complexity with regard to checking tools against geometric constraints, such as feature dimensions and inter-feature clearances. Only the most critical tool dimensions are considered. For example, a drill's length and diameter is compared with the diameter and depth of a cylindrical thru-hole. This is the third major simplification.

The feasibility assessment, as implemented, has some significant limitations. The current capability of the feasibility assessment program is limited to 1D-orientation form features, but this domain includes a respectable fraction of machined parts, because 1D-orientation features are the easiest to produce with three-axis(or less) machines and are thus preferred whenever possible.

This development does not consider 3D form features, but that does not mean that the general methodology could not be applied to these types of features. If such an extension were to be made, one would need to reevaluate the development in order to identify all areas where revisions would be required. What has already been learned by implementing the current development would make extension easier. Extending the producibility evaluation method would be easier than extending the feasibility assessment method, because the evaluation method is simpler and more modular.

This method is flexible with respect to new machines and tool assemblies by virtue of the modular structure of the FMT tables and the manufacturing resource data. If a new type of machine or tool were introduced, one would merely need to add it to the data structure as a new class. If this new class had any previously undefined attributes, they would need to be accounted for in the applications programs for feasibility and producibility. The FMT tables could be reviewed to determine which features the new machine and/or tool class can fabricate.

The producibility index captures difficulty associated with material selection, part size, part weight, and individual form features. The cumulative effect of all form

features is used in computing the index, but knowledge regarding the contribution to difficulty made by individual attributes of a single form feature is readily available. This index is simple, yet manages to capture overall producibility, as well as feature specific producibility, with a relatively small set of parameters.

D6.2 Recommendations

This method can be improved. Now that the method has been implemented and tested, opportunities for improvement have become evident. These opportunities for improvement are discussed next. The producibility index algorithm is completely modular and can accept any type of feature defined by the system and is essentially more capable than the technical feasibility algorithm; however, the index is only valid if it is computed for a feasible part. This makes computation of the index reliant upon the feasibility checking capability. Improving the feasibility assessment method is the key to improving the overall evaluation method. Initial improvements should be limited to within the scope presented in this thesis. These improvements would include improved modularity, refinement (if needed) and implementation of the feature interference test, and extension of the implementation to include other than 1D-oriented features. Methods of shortening the feasibility assessment time should also be considered. One possibility for shortening the feasibility assessment time is to ignore simple features such as steps and cylindrical holes, as long as the minimum tolerance and surface specifications are above a predetermined level. Many part designs possess large numbers of simple features that may not need to be checked, such as Part4 of the test examples, which is composed primarily of cylindrical holes. The form-feature interference test probably could not be streamlined in this manner, because relative locations and orientations of the features, and not their basic geometry/topology cause interference.

While modularity was a goal during development of the feasibility assessment method, modularity was not achieved in the travel test, nor in either the clearance test and tool dimension test. All three tests are concerned with feature dimensions. The diversity in capabilities of machines and tools with respect to various feature types, as well as the variation in the nature of the critical dimensions from feature to feature, result in numerous special cases and exceptions. Note that the corner radius test is highly modularized, even though it is a dimension test, because the parameter for each feature is always corner radius. In addition, there are two possible tool dimensions to check; cutter radius or edge radius. Modularizing the rules for the clearance, tool dimension and travel tests would be a significant achievement, and should be given serious attention. Based on the experience gained from this research, it appears that improved modularity can result in a loss of certainty, because modularity implies simplicity and simplicity implies a loss of details. Without detailed information there is less certainty. The quality of available information is an essential element for any decision making activity. This would result in less accurate feasibility assessments. This is not to say that some balance can not be achieved, but rather to advise caution. One way to make the feasibility assessment more modular may be to develop small sets of simple rules for each assessment parameter. Each individual rule should be simple and

not specific to any particular machine, tool or feature class, so that feature-machine-tool combinations could be evaluated by basically the same rules. Each rule set should address the same parameter, but in an independent manner, so that the multiple tests form a composite test that is sufficiently comprehensive. The rules within a set would need to be written and linked such that all test conditions must pass for the part design to be positive. The structure would be more analogous to an anti-aircraft cannon than a net, where the rules are the projectiles and the aircraft is the part design's feasibility. If one projectile hits the target the aircraft is brought down. The important concept to emphasize at this point is that more than one independent rule for assessing the same feasibility parameter is needed for the purpose of improving the certainty of the assessment, because of the loss of detail inherent in a simple generic rule.

The next step towards enhancement of this method should be to add 2D-orientation and 3D-orientation feature assessment capability. One would need to study the interactions of these features with one another to see the effect on feasibility. For example, assuming a clear tool approach path, one may wish to know the minimum feasible angle between a cylindrical hole and its contact face. Implementation of the feature interference algorithm is more desirable if 2D & 3D-orientation features are added.

This method may be expanded to include other classes of form features and machine tools. For example, milling machines with more than three-axis capability. This action would take the development beyond the present scope of research; for example, adding a new tool class would be necessary. The expansion would be aided by the knowledge and experience gained from the current development.

Adding custom tool design capability is one option for extending the capability of the system, and thus improving the producibility evaluation method. The challenge with regard to designing custom EDM tools is that they require their own design efforts that may include all the activities required for the original part design. This added activity may lead to a dead end if the EDM tool design is not technically feasible. Such potential complications require careful consideration, before a serious development effort is made.

The recommendations for improving the method development and implementation are all expected to make the system more capable, especially with regard to the range of application. Recall that standard (off-shelf) tool assemblies and general-purpose machines were specified in the scope of the development. This does not necessarily mean that the implemented system is applicable to general use. The system is only capable of dealing with processes that are within the range of capability of the specified equipment. General-purpose machines are merely machines that are not specialized, such as mill versus gear hobbing machine, and does not imply that they can do everything. By adding new classes of machines, tools and features, the method will be extended more in the direction of general or universal utility.

The discussion up to this point has been limited to the machining process, but this method is generic enough in the structure of the framework that it could be developed for other processes. The producibility index for a part design with respect to one process could be integrated with those for other processes for compiling a process-

comprehensive producibility index. A weighing scheme would have to be developed so that the relative impact of each contributing process on the part design's producibility could be captured.

References

1. Gopalakrishnam, B. and Pathak, M., "Machine Parameter Selection And Cost Estimation Techniques: Applications In Concurrent Engineering", *Proceedings Of Manufacturing International*, ASME, 1992, pp. 249-256.

2. Bala, M. and Chang, T., "Automatic Cutter Selection and Optimal Cutter Path Generation for Prismatic Parts", *Advances in Manufacturing System Engineering*,1988, pp. 57-68.

3. Ranyak, P. and Fridshal, R., "Features For Tolerancing A Solid Model", *Computers in Mechanical Engineering*, ASME, July, 1989.

4. Sanchez-Garcia, J., "Expert System for Design for Producibility", Ph. D. Dissertation, University of Texas, Arlington, 1989.

5. *Tool and Manufacturing Engineer's Handbook, volume 1*, SME, Dearborn, MI, 1983, pp. 1-49 through 1-51.

6. Chen, C. and Wu, J., "Feature Modeling For CAD/CAM Systems Integration", *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing*, Singapore, 1993, volume 2, pp. 1009-1015.

7. Chen, C., Swift, F., Lee, S., Ege, R. and Shen, Q., "Development of a Feature-Based and Object Oriented Concurrent Engineering System", *Journal of Intelligent Manufacturing*, 1994, 5, pp. 23-31.

8. Waters, T. et al, "Revised NIOSH Equation for the Design and Evaluation of Manual Lifting Tasks", *Ergonomics*, 1993, 7, pp. 749-776.

9. *Tool and Manufacturing Engineer's Handbook, volume 1*, SME, Dearborn, MI, 1983, pp. 2-1 through 2-4.

10. *Tool and Manufacturing Engineer's Handbook, volume 1*, SME, Dearborn, MI, 1983, pp. 1-21 through 1-27.

Appendix E: Process Planner

Appendix E : Process Planner

This system has been developed to handle prismatic part designs with an overall cuboid part shape. Its planning capability is limited to the form features which can be modelled with the product modeling system developed in this project. The system is capable of handling 2.5D form features using machines having up to 3 axes. The planning system provides intermediate shape information of the stock at each machining stage, defines the material removal volume at operation level and relates each form feature to a feasible machine and tool instance combination. It also provides a means to search and optimize the process plan. This system does not, however, evaluate fixtures or generate setup configuration.

E1. Process Plan Class Structure

A hierarchical object class structure to organize a process plan is shown in Figure E1. The structure organizes required machining activities and MRVs in a hierarchy. The process plan arranges the required machining activities into sub-processes and assembles MRVs into MRV subsets. Each subset corresponds to a subprocess. A sub-process has a part setup and a number of operation clusters. A part setup specifies a machine instance and a fixture assembly with setup instructions to be used during the subprocess. It also associates with an NC program which contains the machine code to run the machine and has a set of machining parameters used in this subprocess. The NC program, machining parameters and machining code are specified by the NC Program generator. An MRV subset consists of a number of MRVs to be processed on the same machine in the same setup. Each operation cluster corresponds to a single MRV and records the operations required to remove the entire MRV. An operation is the elemental machining activity toward the removal of an MRV using exactly one cutting tool. The material removed by an operation is called an elemental MRV. Consequently, an MRV is a collection of elemental MRVs, which number is equal to the number of the operations in the corresponding operation cluster. An MRV may have one or multiple islands, which are located within the MRV and are considered as a physical constraint to cutting tool access.

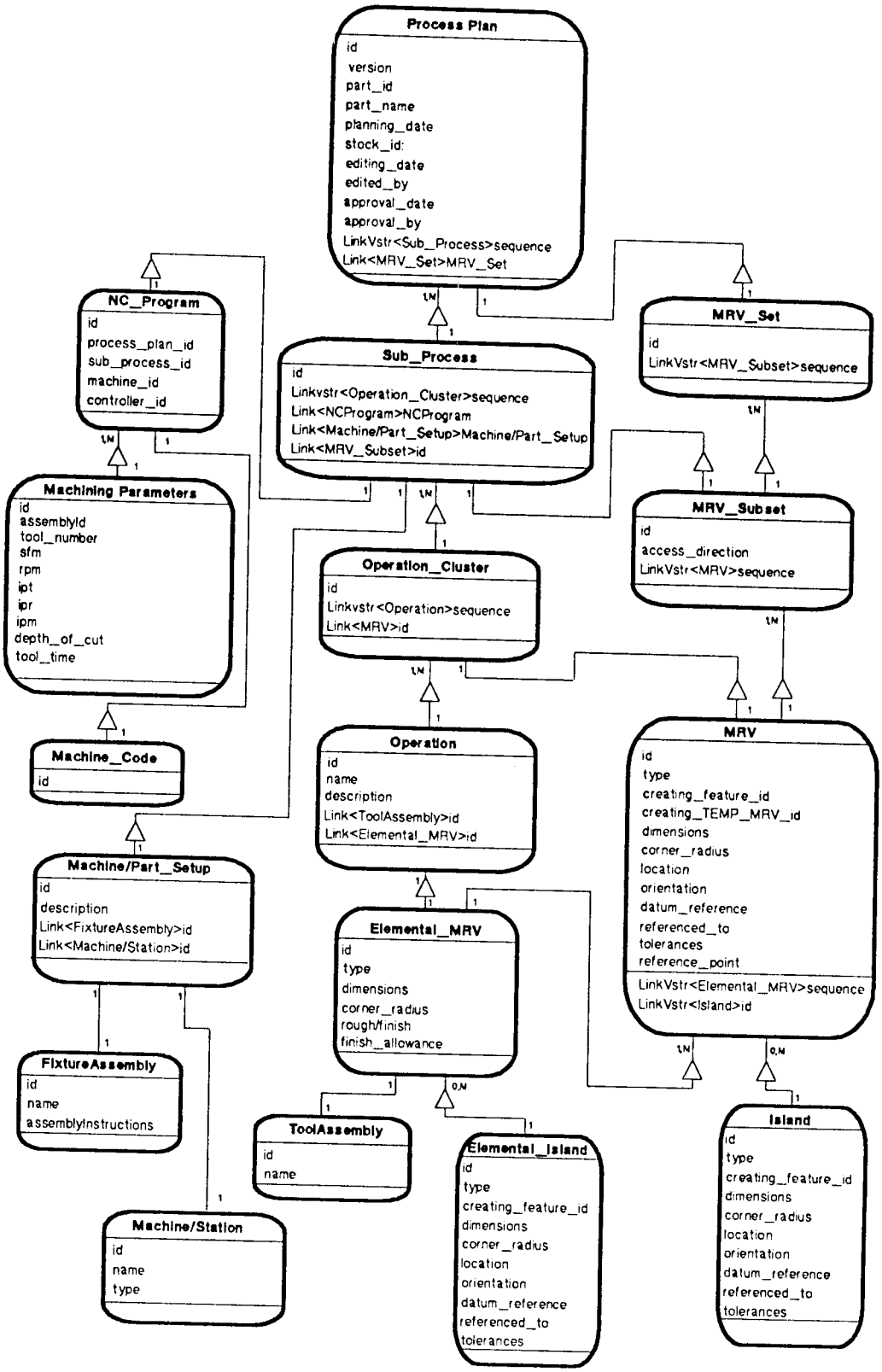


Figure E1: Process Planning Class Structure

E2. Process Planning Methodology

As illustrated in Figure E2, the planning methodology consists of three major phases. In phase one, the user interfaces with the system to select a product model and a planning objective. The product model is retrieved and form features are converted into a sequence of material removal volumes (MRVs). A material stock instance is then selected from available resources and the additional material between the selected stock and the part's bounding box is transformed into slab MRVs. In phase 2, a feasible machine and its required tool(s) are selected for processing of each MRV, while attempting to minimize the number of required workpiece setups.

In phase three, alternative plans are generated and evaluated to identify the optimal process plan according to the specified planning objective. The procedure includes two feedback loops to search for alternative plans. During a feedback loop, alternative machine and tool instances are searched for prior to generating an alternate MRV set. The search for alternative plans continues until a stopping criterion is met. Stopping criteria include elapsed computer run time, a preset number of alternative plans, a planning objective threshold, or a complete search. At the end, the user is presented with the best process plan(s) for review and editing. Upon user approval, the best plan is stored in the database according to the proposed process plan model.

E2.1 Feature Conversion/Objective & Stock Selection (Phase 1)

This planning phase consists of three steps: (1) selection of planning objective, (2) feature conversion, and (3) stock selection. These steps are detailed in the following subsections.

E2.1.1 Planning Objective Selection

The selection of a planning objective is the first step in the first planning phase. The objective is used to evaluate alternative plans and to select the best plan. Typical planning objectives include minimum number of workpiece setups, minimum setup time, minimum total machining time, and minimum total machining cost. The current implementation provides two objectives for selection: minimum number of setups and minimum machining cost. The number of setups resulted from a process plan is tallied once required machines and tools are selected for a design. The machining cost is estimated by adding relative cost factors associated with

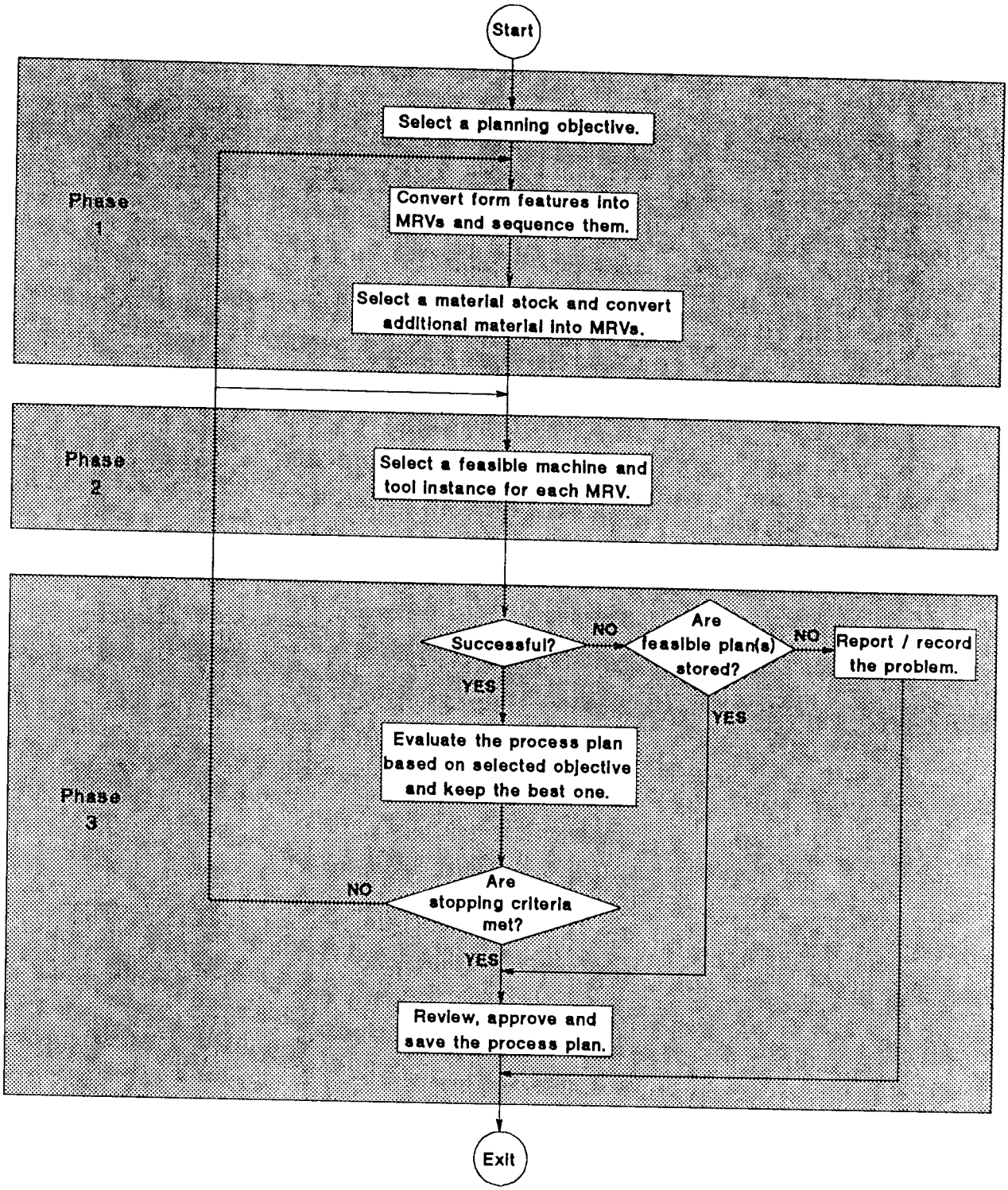


Figure E2: Process Planning Methodology

each machine used to produce the part. These objectives are used to evaluate alternative plans and select the best in the third planning phase.

E2.1.2 Form Feature Conversion

Regardless whether form features are extracted from or stored explicitly in a CAD model, they are design features, which are not necessarily manufacturing features. In fact, manufacturing features are generally defined (interpretable) only within an intended manufacturing process domain. Form features defined for two manufacturing processes could be extremely different. Among various manufacturing processes, the material removal process has been most frequently studied and reported. From a machining point of view, a manufacturing form feature is a material removal volume (MRV) defined with machining related data such as shape, dimensions, orientation, and tolerances. An MRV should be specified to support all machining planning purposes such as operations selection, fixturing, NC coding, and inspection. A material removal volume may be further decomposed into a sequence of elemental material removal volumes for technical and/or economic reasons.

From the machining point of view, there are two general types of design features: protrusion and depression. The conversion of a depression to an MRV appears to be relatively straight forward, because the shape of a depression feature in general can be conveniently related to the shape of a material removal volume. Its volume and orientation could also be precisely transferred from the CAD model. The conversion of a protrusion feature, however, is much less straight forward, because a design feature in this case does not present an enclosed volume for material removal, rather it only defines a spatial constraint to the overall design. The corresponding removal volume has to be jointly defined by the protrusion and other possibly more dominating factors such as stock selection and machine tools to be used, because they actually prescribe the overall shape and size of the feature(s) to be machined. In addition, adjacent design features may further complicate the conversion.

There are two general approaches to feature modeling. One approach is *design by features*, with which the user models product designs with a set of pre-defined form features. The main problem with this approach is the limitation of available form features. A variation of this approach is called *design with features*, which allows the user to model a product design

with both pre-defined form features and conventional geometric entities such as CSG solids [1,2]. This approach tends to attach a separate feature graph to the geometric model of the design. It may face a difficulty in editing form features. For example, a pocket feature instance, after being dragged through the material, should have become a through-hole; but the feature may be still registered as a pocket.

The other feature-related modeling approach has been reported in, for example, Requicha and Chan [3] and Chen and Wu [4]. This approach retains the use of CSG input, while keeping track of all modeling data (both CSG and Brep) in a feature model for feature recognition, editing, and analysis. The advantages of this approach are: 1) the modeling flexibility of a conventional CSG approach and 2) its support for active feature editing and modification. As detailed in Chen and Wu [4], the recognition of a feature with this approach requires only naming the feature; all feature data and attributes are specified and compiled at each modeling step, regardless whether a feature name has been designated. The feature naming activity can be performed at the end of a modeling session or after each geometric computation. The development of the proposed feature conversion algorithm is based on the product model presented in Wu and Chen [5]. The product modeling system reported in Wu [6] will be used to create product designs for testing the proposed conversion algorithm. A product model created with this modeling system is presented in Figure E3. A brief description of this model and the modeling system is given below.

The background in this figure is an X-window screen of the product modeler, which models a product design with eight function menus. Also shown in this figure are: 1) an isometric view presentation of the part design, 2) a feature graph summary, and 3) an information summary for a form feature. This example design consists of twelve form features organized into four hierarchical levels. On the top of the graph is the base (part) feature, which has seven child features in this design. A child feature is a form feature which is created later than its parent and shares some geometric entities. For example, the second feature has the first form feature as its parent and the third feature as its child. A feature may have multiple parent and/or child features.

An information summary for the eighth form feature is illustrated in this figure. The information is retrieved from the product model. The feature has an ID (F8) and its type attribute

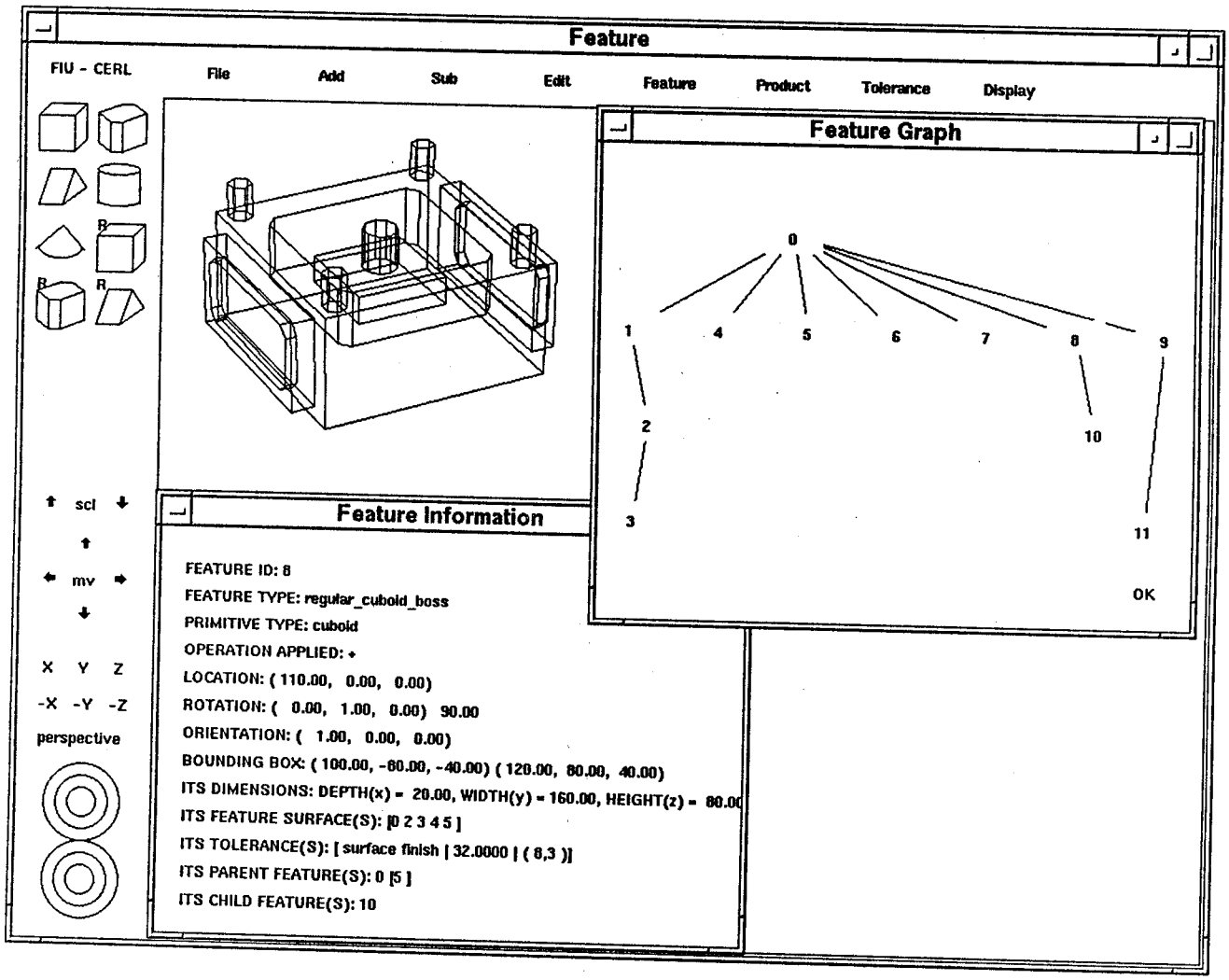


Figure E3: A Sample Part Design

(regular_cuboid_boss). The feature was created with a cuboid solid and located on the fifth face of the base part feature (F0). The information shows that its parent feature is the base feature (F0) and it has a child feature (F10). The feature is located with its center at (110,0,0), rotated 90° about the +Y axis, and oriented in the +X direction. The bounding box of this feature (defined as the minimum cuboid dimensions required to enclose the feature) and its dimensions are both specified as well. The surface of F8 is defined with five faces (f0, f2, f3, f4,and f5) One surface finish of 32μ inch is specified for the third face of this feature.

A feature conversion technique is proposed for converting design features defined in a product model to material removal volumes (MRVs) for machining operations. This conversion process consists of four main steps:

1. Convert each design feature into a temporary MRV (TMRV),
2. Identify the status of each open face on each TMRV,
3. Decompose each TMRV into one or multiple final MRVs (FMRVs), and
4. Sort FMRVs into groups and sequence them.

This part of the appendix focuses on the conversion algorithm developed for prismatic part designs modeled with a cuboid base part feature and thus holding an overall prismatic shape. In the conversion process, design features are retrieved from the product model and evaluated one by one to identify their corresponding MRVs and in the meantime estimate a bounding box of minimum dimensions required to enclose all evaluated form features. The bounding box assumes a cuboid shape because the part design is assumed to have a general cuboid shape; however the shape of the stock material to be selected does not have to be rectangular. In the design process, a cuboid solid is typically created first as the base feature for a prismatic part design, on which other (protrusion and depression) features are gradually added.

The bounding box needs to be re-evaluated and expanded only when a protrusion feature is converted. The expansion may lead to either enlarging an existing MRV or creating a new MRV. In case of creating a new MRV, the base face of the new MRV is defined by the face of the current bounding box on which the feature is located. The height of the MRV is defined by the amount of protrusion beyond the bounding box. The extension direction is typically along the normal vector of the base face for a 2 and 1/2D form feature. The MRV generated from converting a protrusion feature is always a cuboid with island(s) and is called a cuboid_slab in

this paper because five of its six faces are open at the time of creation. The protrusion feature is viewed as a constraint (island) to the newly created MRV.

The conversion is carried out in four steps: (1) temporary manufacturing feature generation, (2) open face status evaluation, (3) manufacturing feature refinement, and (4) manufacturing feature sequencing. They are further detailed below.

E2.1.2.1 Temporary manufacturing feature generation

The first stage is to convert design features into temporary MRVs and is depicted in Figure E4. The conversion procedure can be summarized as follows:

1. The conversion starts with the base part feature and a search pointer is used to pinpoint the current search feature.
2. It ends when the pointer returns to the base feature and all child features of the base feature have been converted.
3. It always searches and converts depressions first, before converting any protrusion on the same feature node.
4. Once a feature (depression or protrusion) is converted, the conversion of all its child features must immediately follow.
5. When the currently pointed feature has no child feature or all of its child features have been converted, the feature pointer moves back to its parent feature.

Following the flow chart in Figure E4, the base feature is first retrieved from the product model and processed. The bounding box is initialized and set to the dimensions of the base feature. The feature search pointer is initialized, pointing to the base feature. The search for child features at the base feature node begins and an unconverted depression feature is selected for conversion, if it exists. After the conversion, the search feature pointer is set to the newly converted feature. The conversion process continues at each feature node, searching and converting first child depressions and then child protrusions. It ends only after the search pointer returns to the base feature and all its child features are converted.

The conversion of a depression feature is straight forward. When a depression is encountered during search, a temporary MRV (TMRV) instance of the same type is created and its attributes are copied from the design feature in the product model. No adjustment to the

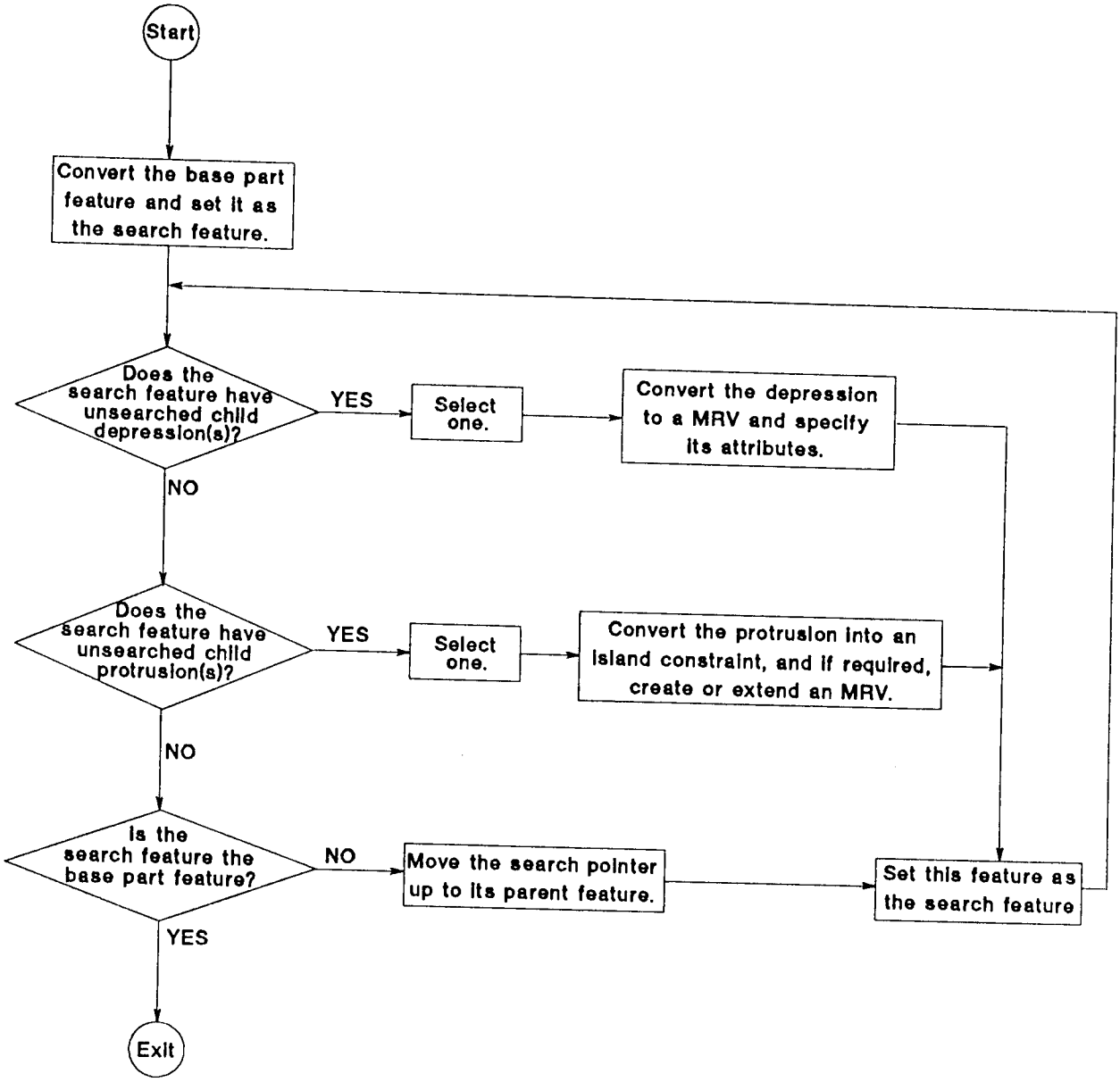


Figure E4: Temporary MRV Generation

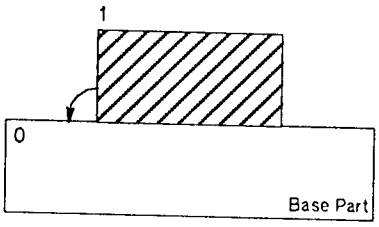
bounding box is needed for converting a depression because no extra material is added to the design by the depression. When a protrusion is converted, however, extra effort is required. Depending on its interaction with other features, converting a protrusion may result in taking one or a combination of the following three actions:

1. Convert the protrusion into an island to constrain an existing MRV.
2. Create a new MRV and convert this protrusion into an island to constrain the new MRV.
3. Extend an existing MRV and convert this protrusion into an island to constrain the extended MRV.

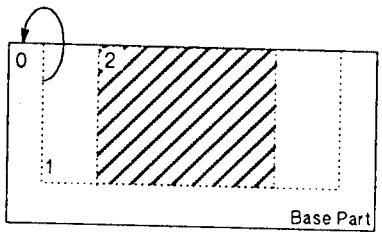
To further explain the conversion of a protrusion, four part designs are illustrated in Figure E5, each depicting a different feature layout possibility. In case (a) where a protrusion is on a face of the base feature, action #2 should be taken if no MRV has been created on this face. If an MRV already exists on this face and it encloses the protrusion entirely, then action #1 should be taken. If the protrusion extends beyond the existing MRV, then action #3 should be taken instead.

The case (b) in the figure illustrates a protrusion being located in a depression (a pocket in this case). Since the boss is inside the pocket, action #1 should be taken to convert the boss into an island to constrain the pocket MRV. This constrained pocket MRV is commonly called a pocket with an island. If the boss extends beyond the pocket, both actions #1 and #2 should be taken. In other words, the boss is first converted to an island to constrain the pocket MRV and then a new MRV is created on the top of the pocket MRV to enclose the additional portion of the boss.

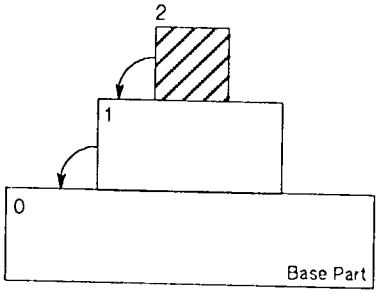
Cases (c) and (d) illustrate a more complicated situation in which the protrusion is on a protrusion. Before any conversion action is taken, a search for the protrusion's ancestry in the feature graph beyond its parent feature is required until a non-protrusion ancestor (either a depression or the base feature) is found. In case (c), the search reaches the base part. The action plan described in case (a) should thus be followed. That is, to create a new MRV, modify an existing MRV, or do both. In case (d), the search stops when reaching a depression, for which an MRV has already been created and whose child protrusion has been processed and linked tothe MRV(s) as a constraint. The action plan for case (b) should then be followed.



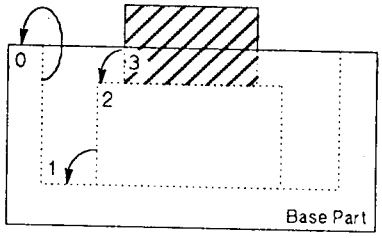
Case (a): Protrusion on Base Part



Case (b): Protrusion in Depression



Case (c): Nested Protrusions on Base Part



Case (d): Nested Protrusions in Depression

Figure E5: Four Protrusion Layout Cases

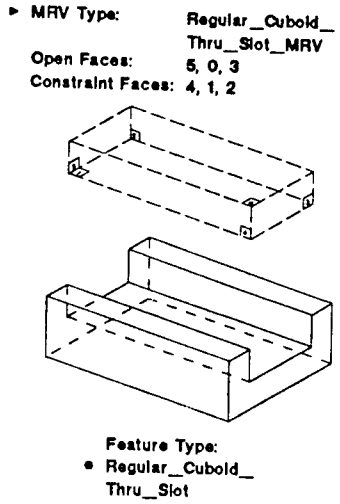
E2.1.2.2 Open Face Status Evaluation

Temporary MRVs are defined with a data structure similar to the one for form features in the product model. However, two additional data have to be included in the MRV definition. They are: open faces status and island constraint. An open face is defined as a face being added to the part design while creating a protrusion, or a virtual face of a form feature which is clipped off from the parent feature while creating a depression. The use of an open face in machining is to capture a feasible tool access direction to the form feature (as well as its corresponding MRV) in order to machine it. The other face type on a feature surface is called a constraint face, which is a real face and represents a physical boundary for tool motion. Both constraint faces and open faces are captured in the product model. However, the status of the open faces in the design model may be tentatively altered, as MRVs are created. The status of faces created on MRVs also needs to be evaluated.

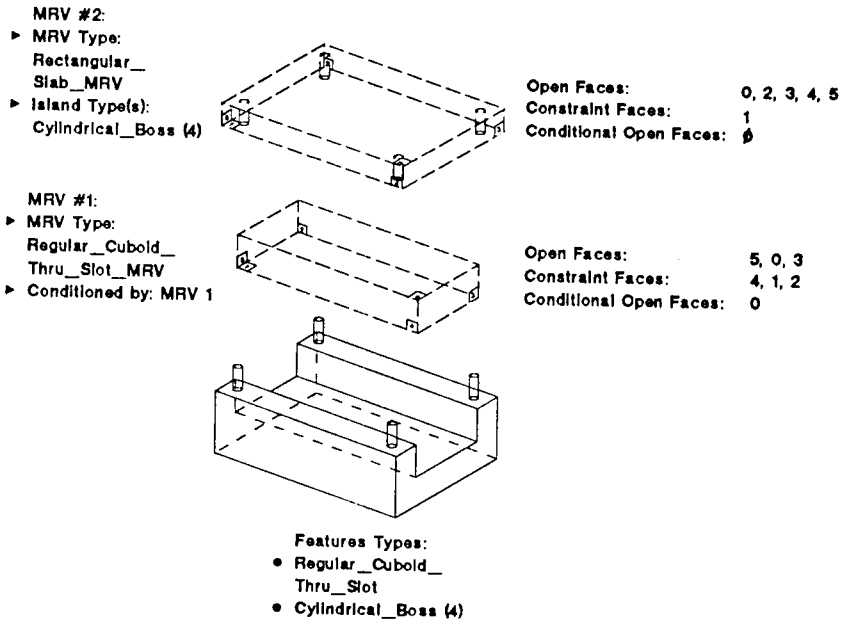
Each time a new MRV instance is created or extended to enclose a protrusion, the MRV may create new open faces and may also tentatively block the access to some open faces on existing MRVs. When this occurs, the status of the affected open faces are identified and labeled as conditionally open. It means the open face is no longer accessible until the blocking MRV is removed (i.e., machined off). As shown in Figure E6(a), the through-slot MRV has three open faces (f0, f3, and f5) and three constraint faces (f1, f2, and f4). When the MRV (#2) is created for the four cylindrical protrusions and placed on the top of the slot MRV (#1) as shown in Figure E6(b), the originally open face (f0) of MRV1 becomes conditionally open and remains conditional until the new MRV (#2) is machined. The other two open faces are not affected by this new MRV. The conditional status of an open face also forms a constraining relationship between MRVs. When the blocking of an MRV by another is detected, an inter-MRV constraining relationship is thus established and has to be exercised later when generating feasible MRV machining sequences. In this example, the removal of MRV1 is feasible only after MRV2 is removed, if the machine access is chosen to be from the +Z direction.

E2.1.2.3 Manufacturing feature refinement

In the previous two stages, all design features in the product model have been converted into TMRVs and each open face status on an MRV has been examined and labeled with either



(a)



(b)

Figure E6: Open Face Status

an open or conditionally open status. The third step is then to examine each temporary MRV and its enclosed island(s). Each MRV having one or more constraining islands is singled out and decomposed into one or more elemental MRVs. An elemental MRV is defined as an MRV which contains either no island or island(s) with the same height as the MRV. The reason for the refinement is to facilitate process planning and NC programming. All MRVs with islands of different height or with an island located on other island(s) need to be evaluated. The general approach is to slice each temporary MRV along the height of each island.

As outlined in Figure E7, the refinement process evaluates each TMRV, decomposes it when needed, and translates each into final MRV(s). It first checks whether a selected temporary MRV has an island. If not, the MRV requires no refinement and a final MRV instance is created to which all of the temporary MRV's attributes are copied. If there exists island(s), the '*ttm*' and the '*btm*' values are calculated for each island and are used to slice the temporary MRV into final (and thinner) MRVs. The '*ttm*' length is defined as the distance from the *t*op of the island *t*o the bottom of the *M*RV to which it is attached. The '*btm*' length is defined as the distance from the bottom of the island to the bottom of the MRV to which it is attached. All the islands enclosed in a temporary MRV are ranked according to their '*ttm*' value for slicing.

In the process of slicing a temporary MRV, the island with the currently highest ranked *ttm* is first identified and selected. If the height of the temporary MRV, denoted as *H*, is greater than the *ttm* value of the selected island, a final MRV with no island is created and its thickness is set to be the difference between the *H* and the *ttm* values. After the dimensions and location are specified for the final MRV, the height of the remaining temporary MRV is adjusted to the *ttm* value of the selected island. The slicing process repeats for the remaining portion of the temporary MRV with the newly adjusted height (H).

On the other hand, if the *H* value is equal (or less) than the *ttm* value of the selected island, a final MRV with island(s) is to be created. To create a final MRV in this case, the island with the second highest *ttm* value (less than the currently highest *ttm*) needs to be retrieved. The difference between the two *ttm* values is calculated and set to be the thickness of the final MRV being created. After its location and other attributes are specified, each island (partially or entirely) inside the final MRV is computed and attached to it. Before repeating the

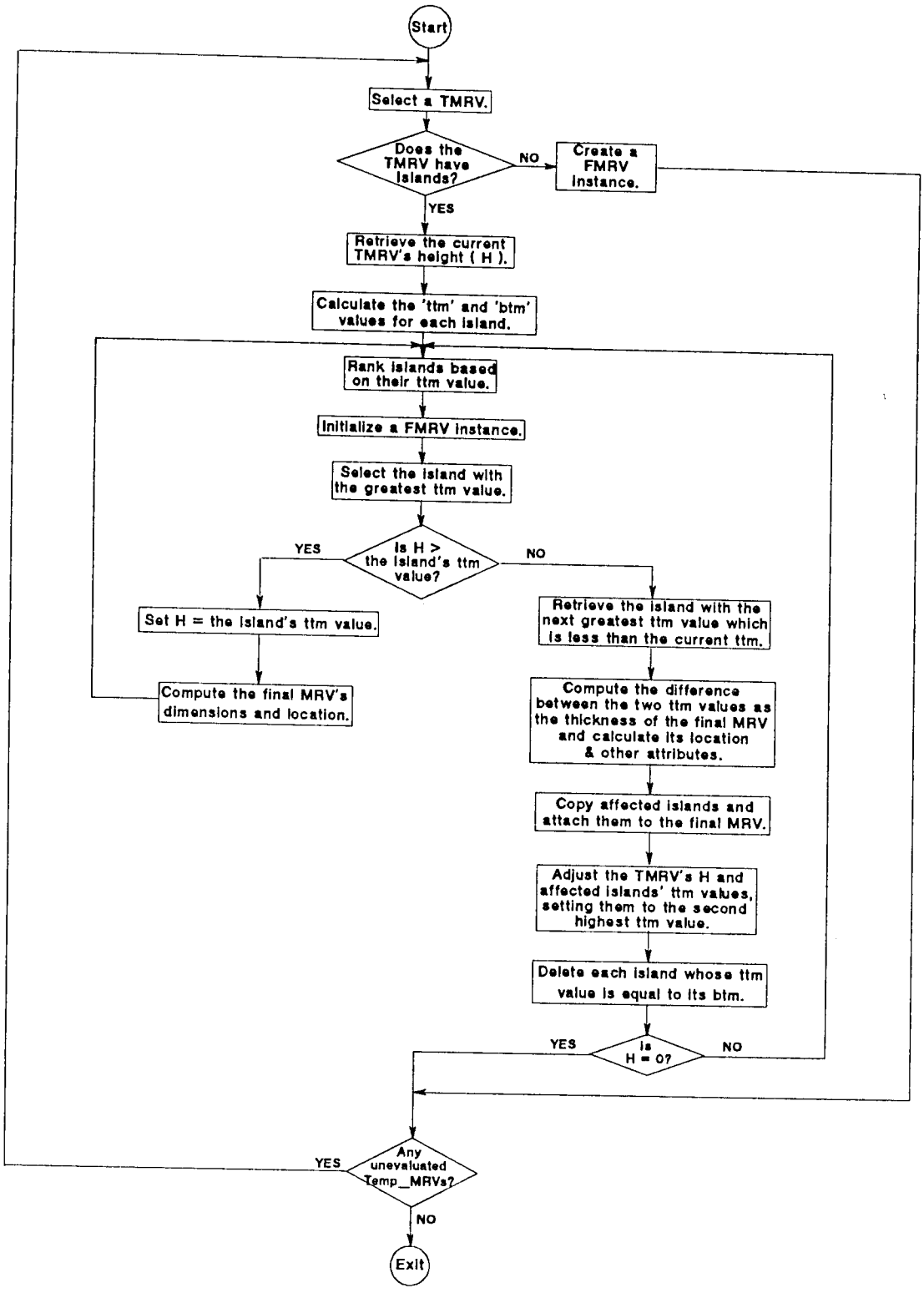


Figure E7: Temporary MRV Refinement

slicing process, the *H* value of the current temporary MRV and the *ttm* value of the affected island(s) are adjusted by setting them to the currently second highest *ttm* value. Also after the adjustment, any island in the temporary MRV whose *ttm* value is equal to its *btm* value is eliminated from further evaluation. The refinement of a temporary MRV is completed when its current *H* value reaches 0. The entire refinement process though continues until all temporary MRVs are examined. In decomposing a temporary MRV, open faces and their status are progressively examined and labeled on the decomposed final MRVs accordingly. The constraining status between two temporary MRVs is also re-evaluated and updated for their corresponding final MRVs.

Figure E8 shows a part design which has three bosses and a pocket. One boss is located inside the pocket. During the first stage, two temporary MRVs were created: a slab MRV with two islands and a pocket MRV with one island. Since the pocket is deeper than the height of its enclosed island, the pocket MRV needs to be decomposed into elemental MRVs: one with no island constraint and the other with an island whose height equals to that of its enclosing elemental MRV.

E2.1.2.4 Manufacturing feature sequencing

The last conversion stage sorts final MRVs into MRV groups based on the status of their open faces. The grouping procedure observes the following guidelines:

1. Each MRV must be assigned to an MRV group.
2. MRVs in each group must have the same orientation.
3. A constrained MRV can be assigned to a group only after all its constraining MRV(s) have been grouped.
4. Final MRVs are pre-sorted by their orientation and MRVs in each pre-sorted group are pre-sequenced based on their face constraints on one another.

Each MRV may have several open faces and therefore in theory may have multiple grouping orientations. To avoid assigning an MRV to multiple groups, only the major orientation of each MRV is used in this development for simplicity. For an application involving three axis machines, the major orientation of a feature is defined along the ±X, ±Y, and ±Z axes. MRVs sharing a slanted face are also allowed to form an MRV group. Both MRV groups and the

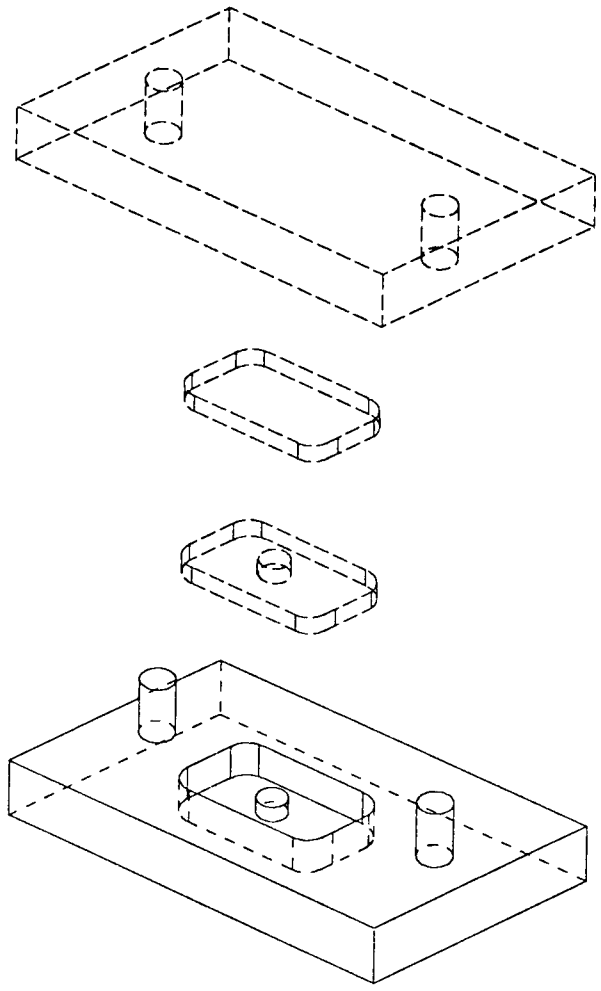


Figure E8: An Illustration of MRV Refinement

MRVs in each group need to be sequenced according to the status of their open face(s). The proposed grouping procedure attempts to concurrently group and sequence MRVs. It begins with sorting final MRVs by their major orientation and then sequencing the MRVs in each preliminary group according to their face constraints on one another (from the least constrained to the most). The rest of the grouping procedure is outlined in Figure E9. It begins by initializing an MRV group with an ID assigned. Initially this group is empty and its orientation is undefined. To start the assignment, an ungrouped MRV is randomly selected. If the MRV has no constraining MRV or its constraining MRVs have all been grouped, it is assigned to the group, which then assumes the same orientation. The procedure continues evaluating the rest of the unassigned MRVs for this group with an additional condition of having the same orientation. After all unassigned MRVs are evaluated, a new MRV group is initiated until all MRVs are grouped. The grouped MRVs are stored according to a data structure shown in Figure E10. With this structure, all final MRVs are stored in an MRV set and organized into MRV groups. Each group consists of one or more final MRVs. Each MRV may have island constraint(s).

E2.1.2.5 Application example

This section provides an application example of the proposed conversion technique to the part design shown in Figure E3. As mentioned previously, this design has twelve form features: the base feature, three depressions, and eight protrusions. Some of them are nested features. The application is explained in the following four subsections, parallel to the development presented in Section E2.1.2.

E2.1.2.5.1 Initial Feature Conversion

The conversion of design features into temporary MRVs is illustrated in Figure E11 in a step by step manner. The left column of the figure shows a feasible feature conversion sequence selected by the proposed algorithm. The feature being processed is denoted with a dotted circle. The middle column of the figure shows a 2-D profile view of the part design with markers designating the feature being processed. The right column presents the result of the conversion as it progresses along the feature graph.

The conversion process begins by setting the base feature (F0) as the search feature and

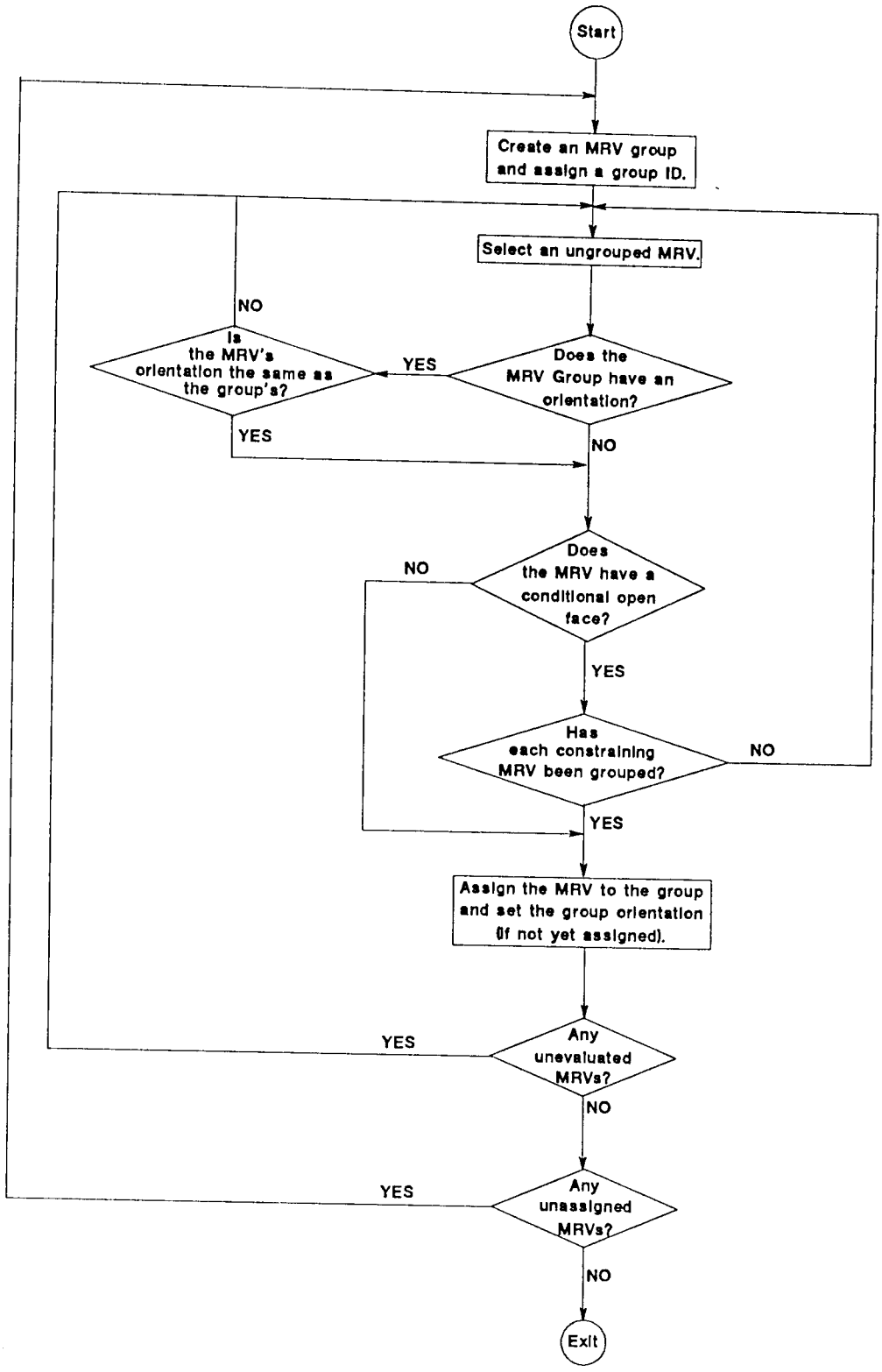


Figure E9: MRV Grouping Procedure

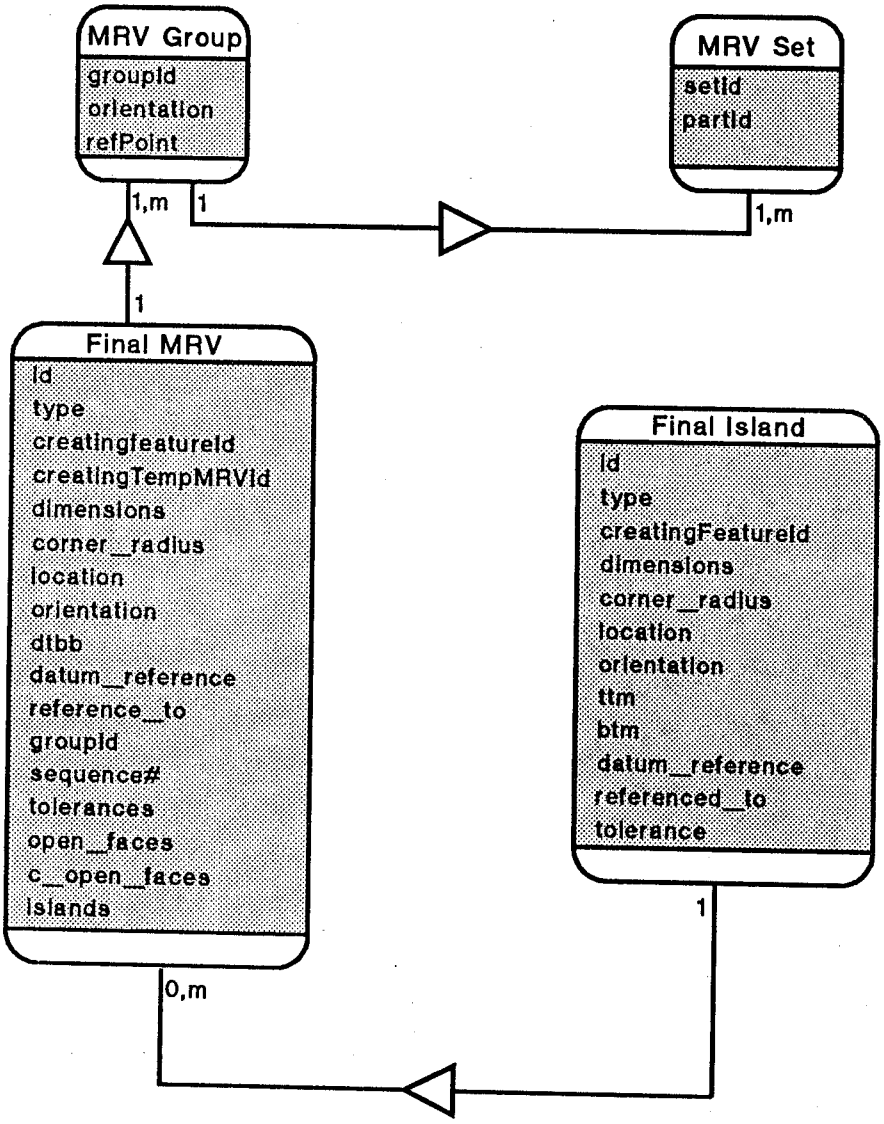


Figure E10: MRV Object Class Structure

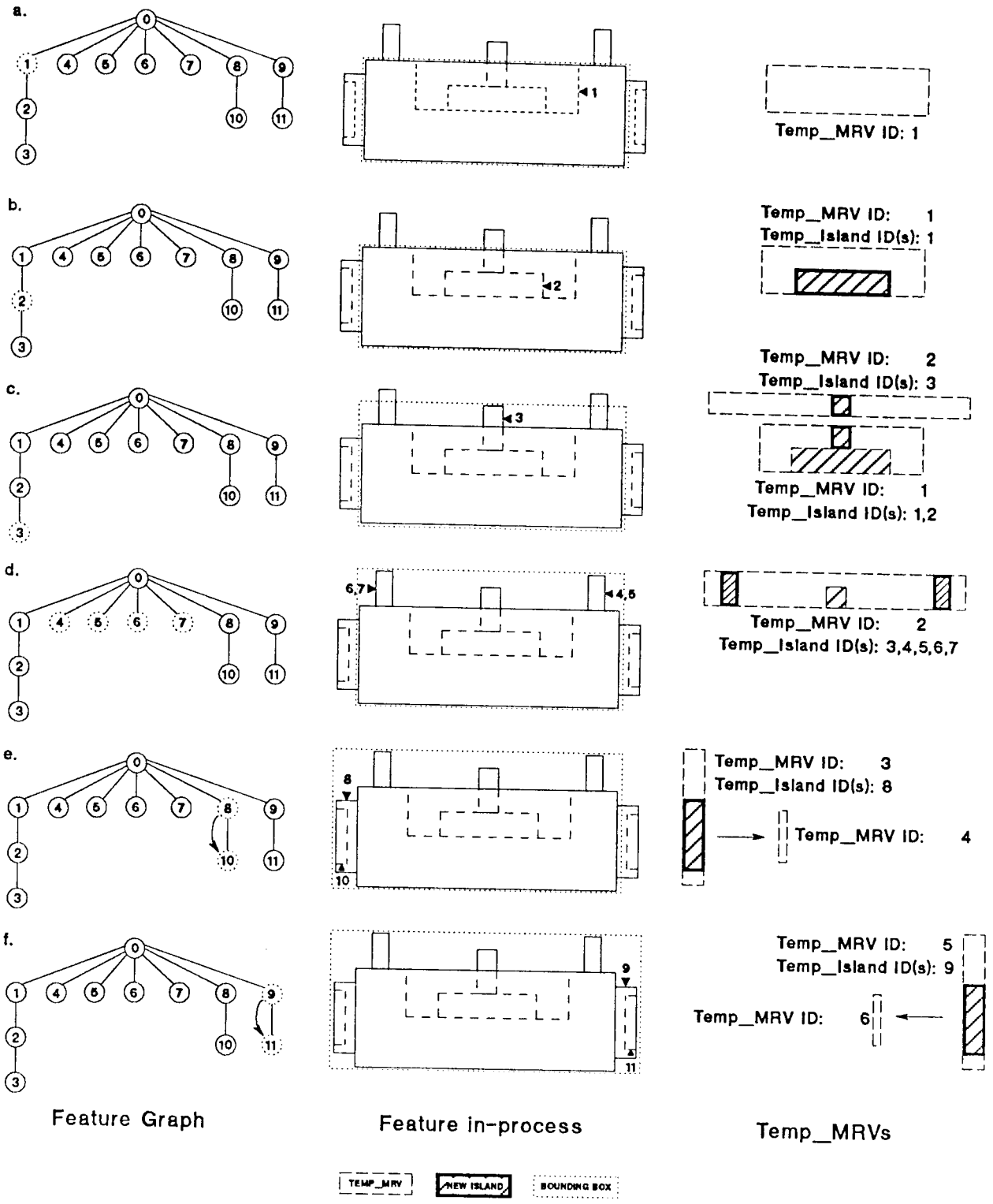


Figure E11: A Feature Conversion Example

creating the initial bounding box with the size and shape of the base feature. A search for a child depression is then launched. Although three depressions (F1, F10, and F11) exist on the design, the base part feature has only one child depression feature (F1). Therefore as shown in Figure E11(a), this rounded_cuboid_pocket feature is selected for conversion, and converted into a temporary MRV, denoted as TMRV1. The search pointer is thus moved to this depression feature and the search for its child depression features on this node is launched. Since this search feature does not have child depressions, child protrusions are then searched. One child protrusion feature (F2) is identified on this node as shown in Figure E11(b) and is therefore selected for conversion. Because the protrusion does not extend beyond TMRV1, no additional temporary MRV is created and this protrusion feature is converted into a temporary island, denoted as TI1, and attached to TMRV1 as its constraining island.

The search pointer is now moved to the protrusion (F2), and its only child feature (F3) as shown in Figure E11(c) is thus selected for conversion. This protrusion feature locates in TMRV1 but extends beyond the MRV. Therefore a new temporary MRV is required to enclose the portion of the protrusion which goes beyond TMRV1. The current bounding box is extended accordingly. The conversion task is carried out in three steps. The first step is to slice the protrusion and create a temporary island (TI2), capturing the protrusion's portion contained inside TMRV1. The second step is to create a new temporary MRV, denoted as TMRV2, by projecting the face of the base feature having the same orientation by the amount that the protrusion extends beyond the current bounding box. A third island (TI3) is created and attached to TMRV2 as its constraint. Finally, the current bounding box is expanded to include TMRV2.

The search pointer is now pointing at the feature (F3). Since it has no child feature, the search pointer moves up to its parent feature (F2) to look for unconverted features, and then back to the parent feature (F1). Both F2 and F1 have no other child features, so the search pointer returns to the base feature (F0). Because the base feature does not have any other child depression feature, the search for a child protrusion feature is therefore begun. At this time, the base part has six unconverted child protrusion features. The procedure proceeds to convert these protrusions one at a time.

As shown in Figure E11(d), the protrusions (F4, F5, F6 and F7, all cylindrical_bosses) are next selected and converted. When the feature (F4) is encountered, it causes TMRV2 to

grow in the direction along which the protrusion extends beyond the bounding box. The MRV's thickness is modified to account for the additional height of the protrusion (F4), and F4 is then converted into a temporary island (TI4) and attached to TMRV2. Again, the bounding box is adjusted to reflect the expansion of TMRV2. The conversions of F5, F6, and F7 do not create or expand any MRV, because they do not further extend the current bounding box. Thus, these three design features are converted into three temporary islands (TI5, TI6 and TI7) and attached to TMRV2. None of the four features has child features, thus the search pointer moves to the base feature each time after conversion.

As the search progresses, the protrusion feature (F8, a regular_cuboid_boss) is next selected for conversion as shown in Figure E11(e). As this design feature extends beyond the current boxing box, the bounding box is expanded and a new temporary MRV, denoted as TMRV3, is created. The thickness of TMRV3 is set equal to the length of the protrusion's extension beyond the bounding box. The remaining dimensions (length, width) are specified such that the MRV completely fills the face of the current bounding box being used for the extension. In other words, the bounding box always expands in one direction at a time. After TMRV3 is created, this protrusion is converted into an island, denoted as TI8, and attached to the TMRV as a constraint. Now the search pointer is set at the feature (F8) and the algorithm begins to search its child features for conversion. The only child depression (F10, a rounded_cuboid_pocket) is thus selected, as indicated in Figure E11(e). For this pocket, a temporary MRV is created and denoted as TMRV4. Since F10 has no child and F8 has no other unsearched child feature, the search pointer again returns to the base feature.

At the base part, the only child feature left unconverted is the protrusion feature (F9) and is therefore selected as shown in Figure E11(f) for conversion. The conversion process is the same as for F8 and a temporary MRV (MRV5) is created. The protrusion is converted into an island (TI9) and attached to the new MRV. Finally its only child feature (F11) is selected and converted into an MRV (TMRV6). The search pointer moves up from F11, then to F9, and finally back to the base feature again. Because all child features on the base feature node have been converted, the conversion process is finished.

E2.1.2.5.2 Open Face Evaluation

The status of an open face can be altered if there is an MRV obstructing it. Therefore, an open face being blocked needs to be re-classified. As each temporary MRV is created or expanded, each of its open faces is evaluated to identify possible blockage. When a blockage is detected, the status of the open face is changed to conditionally open and attached together with the ID of the blocking MRV. The open face evaluation approach for MRVs generated from a depression differs to the one for MRVs generated from a protrusion. Figure E12 illustrates how an open face status is evaluated.

TMRV1 is a depression-converted MRV. Its parent feature is the base part feature and it has an open face with face normal along (0,0,1). Because the parent is the base feature, a search is conducted for TMRVs having the same orientation as the open face. Since TMRV2 is detected with this orientation, this open face status is modified into conditionally open and the ID of TMRV2 is recorded. That is, TMRV1 is accessible from (0,0,1), only after TMRV2 is removed. If its parent feature had been a depression (i.e. two nested depressions), the ID of the temporary MRV generated from the parent feature would have been returned as its constraining MRV. If its parent feature had been a protrusion, a test would have been needed to check if the protrusion originated a temporary MRV. If so, the ID of the protrusion-generated TMRV would have been returned as the constraining TMRV. Otherwise, the temporary MRV, which the protrusion-turned island were constraining, would have been returned as the constraining TMRV.

A protrusion-converted MRV may be created on any face of the base part feature, depending on the orientation of the protrusion on the part design. Both TMRV2 and TMRV3 are protrusion-converted. TMRV2 is created on the zeroeth face of the base feature and has an orientation of (0,0,1). TMRV3 is, then, created on the fifth face of the base feature and has an orientation of (1,0,0). With the addition of a new MRV, all existing MRVs should be examined. In the case of TMRV2, the orientation of MRV3 is used to match the orientation of all the open faces on TMRV2. The fifth open face on MRV2 is found to have an orientation of (1,0,0), identical to that of the TMRV3's. Therefore the status of this face is modified as conditionally open and attached with TMRV3 as its constraining MRV.

E2.1.2.5.3 MRV Refinement

The refinement procedure needs to be applied to only temporary MRV(s) having island

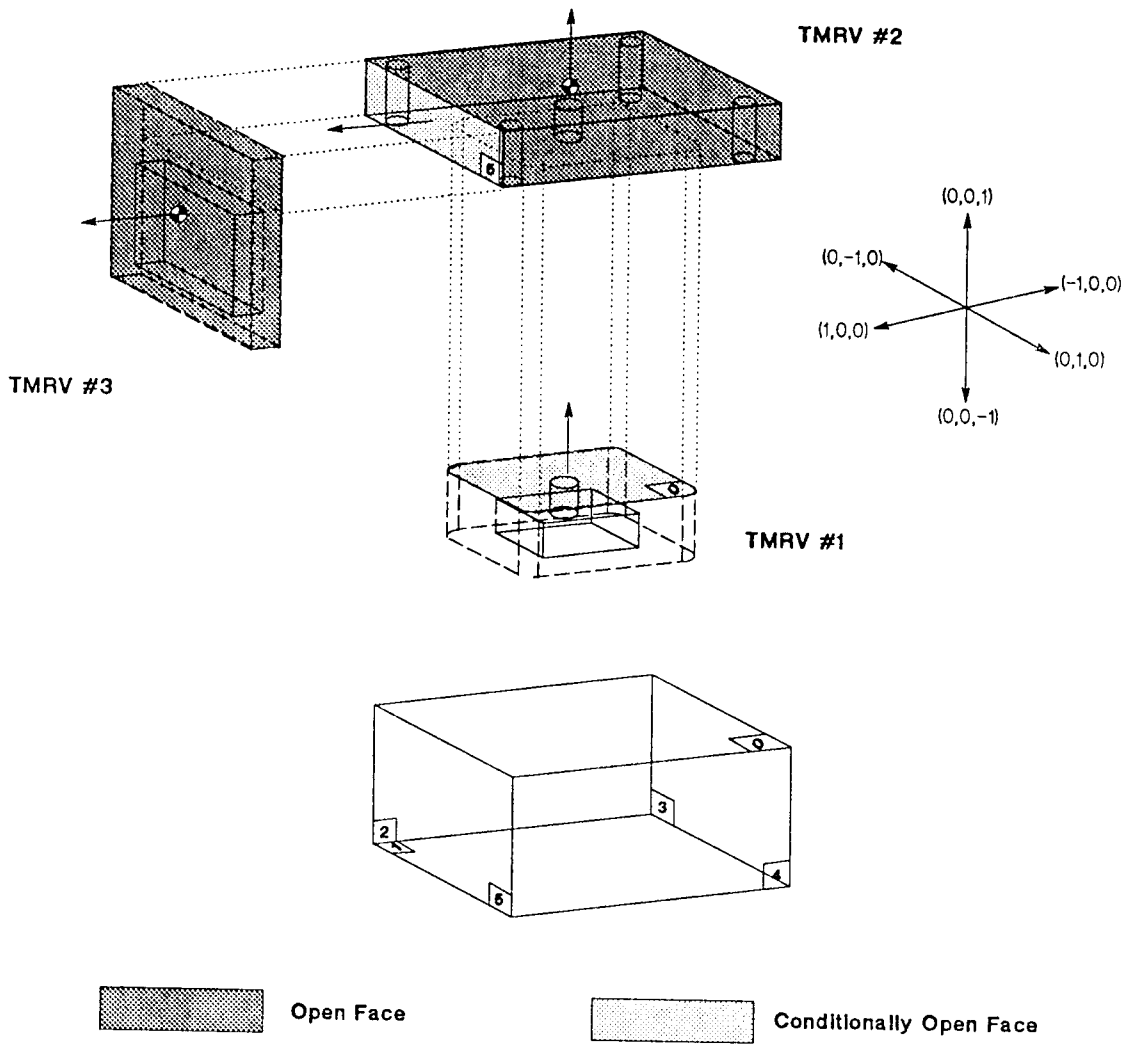


Figure E12: Open Face Evaluation

constraint(s). To decompose an MRV with island(s), all islands inside the MRV are ranked and sliced at each island's height. The refinement of TMRV1 is illustrated in Figure E13. This temporary MRV has two islands of a different height: TI2 and TI3. TI2 is ranked higher because it has a greater *ttm* value. To begin the decomposition, the height (H) of the TMRV is retrieved and compared to the currently highest *ttm* value. Since they are equal in this example, a final MRV instance is initialized and labeled as FMRV1 with expectation of enclosing at least one final island. The thickness of FMRV1 is evaluated as the difference between the *ttm* of TI2 (the currently highest) and the *ttm* of TI3 (the currently second highest). The only temporary island contributing to TMRV1 is TI2 and therefore the final island is created with the data specified in TI2 except that the dimensions and location are modified. This island is labeled as FI1 in the figure and is specified in FMRV1 as a constraint. Before creating the next final MRV, the height (H) of TMRV1 is adjusted with the currently second highest *ttm* value. The *ttm* value of TI2 is also adjusted with the same value, because the island constraint has been evaluated for FMRV1. After the adjustments, the *ttm* and *btm* values of each temporary island are compared. As a result, TI2 is deleted from further consideration because it has equal *ttm* and *btm* values. This means that its role as an island constraint has been fully appreciated. Furthermore the H value after the adjustment is checked. A positive H value implies further refinement is necessary; and therefore the process repeats and the second final MRV is created and denoted as FMRV2 in Figure E13(c). It contains a final island converted from TI3. After the creation, all temporary islands are deleted and H value reaches zero. The refinement process is therefore completed. Figure E14 summarizes the results of the entire feature refinement process for the sample design. The refinement process starts with TMRV2 and yields FMRV3 constrained with four final islands and FMRV4 with five islands as shown in Figure E14(a). The refinement of TMRV1 was detailed in the above paragraphs and generated FMRV1 constrained with one island and FMRV2 with a different island, as shown in Figure E14(b). The remaining temporary MRVs need little refinement because they have either no island in the case of TMRV4 and TMRV6 or an island with the same height as its MRV's. Therefore, their conversion is done by simply creating a final MRV instance and copying all the temporary MRV's attributes to it.

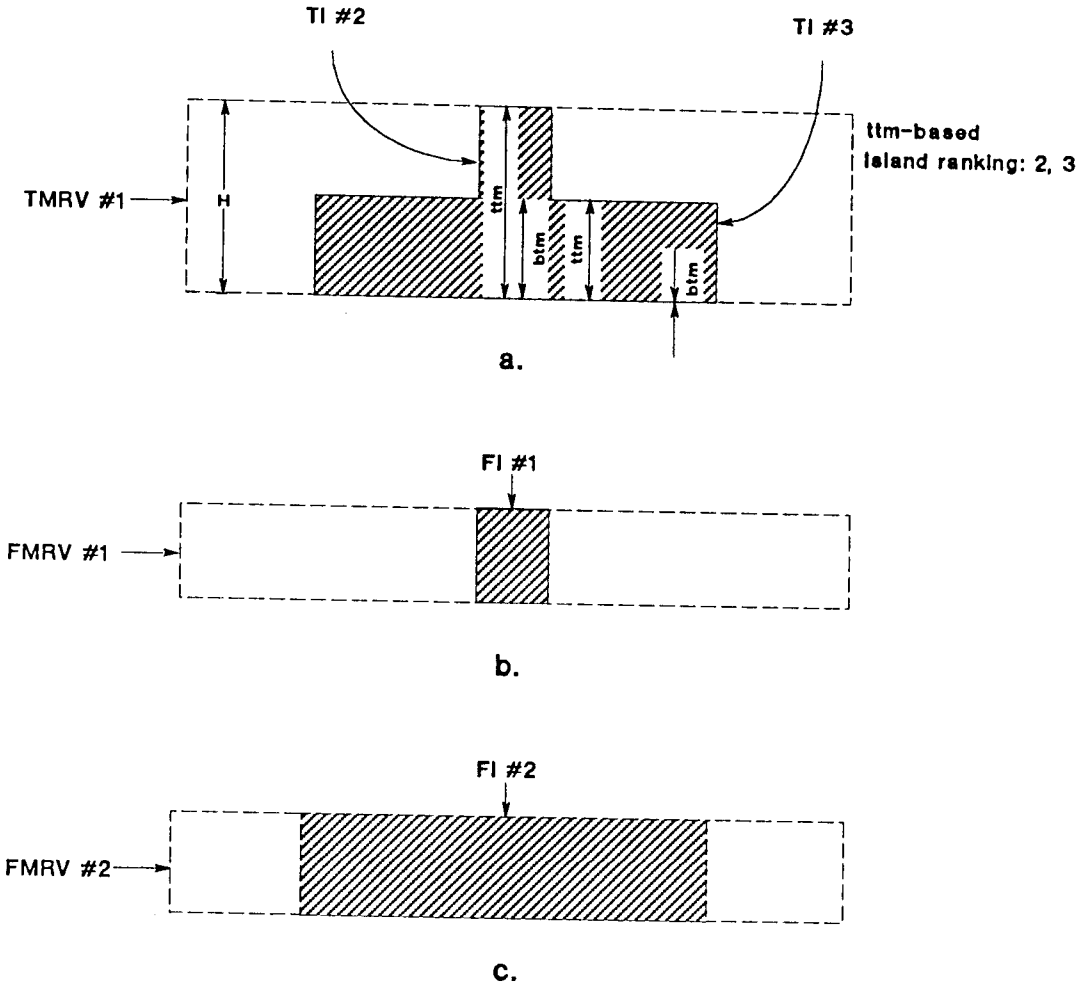


Figure E13: An illustration of MRV refinement

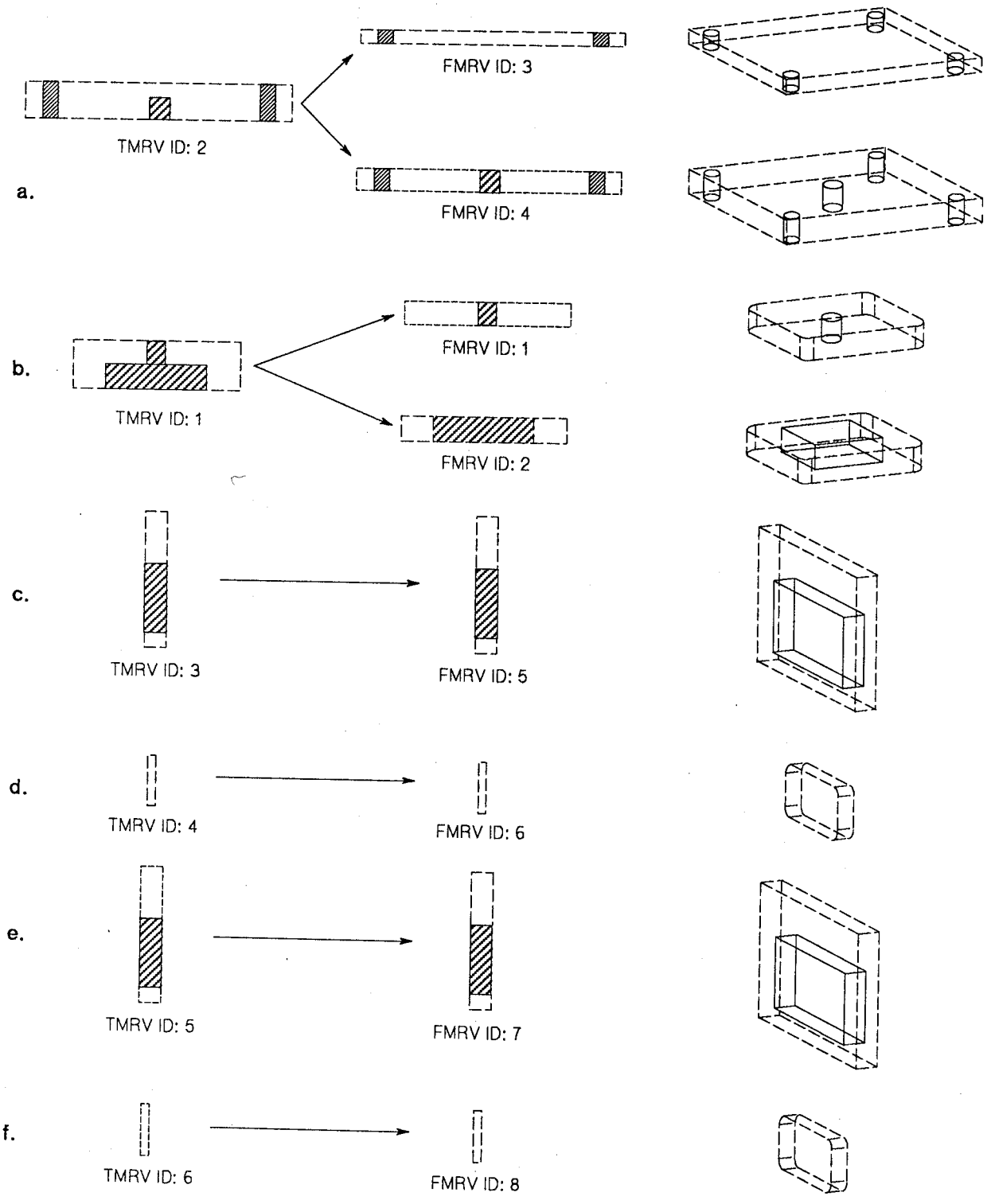


Figure E14: Refinement Result

These final MRVs are then evaluated for their spatial relationship through their open face status. The open and conditionally open faces on each final MRV for this part design are shown in Figure E15. All open faces re shaded dark and conditionally open faces are shaded lightly. Both FMRV5 and FMRV7 in this figure have five open faces, except the bottom face. FMRV6 and FMRV8 both have their top face open. FMRV3 has three (top, front and back) open faces, while its side faces are conditionally open due to the constraint from FMRV5 and FMRV7. FMRV4 has its back (not shown) and front faces open, but its top and side faces are conditionally open due to the existence of FMRVs 3, 5 and 7. Both FMRV1 and FMRV2 have a conditionally open face on the top. FMRV2 is constrained by FMRV1, which is in turn constrained by FMRV4. These face conditions provide a foundation for sequencing final MRVs.

E2.1.2.5.4 Final MRV Grouping

In final MRV grouping, the major orientation of each final MRV is identified and used for preliminary grouping, resulting in three preliminary groups in this example as shown in Figure E16(a). The FMRVs 1-4 belong to a group; FMRV5 and FMRV6 are in one, and the others are in another. In order to increase grouping efficiency, an internal ranking of the MRVs inside each group is also conducted according to their face constraints on one another. In the grouping process, FMRV5 happens to be first selected and an MRV group is initiated. Since it has no constraints and the group has no MRV yet, FMRV5 is therefore assigned to the first final MRV group, denoted as G1, and its orientation becomes the group's orientation. FMRV6, the only other final MRV with the same orientation in this example, is eventually selected and assigned to the same group. After all the ungrouped FMRVs are evaluated, the grouping algorithm repeats with initializing a new MRV group, until all final MRVs are assigned to a group. As a result, three MRV groups are created for this design. The second MRV group is oriented at (-1,0,0) and has FMRV7 and FMRV8. The last group contains the other four FMRVs. During the sequence, the third group is identified with a constraint from G1 and G2 due to their open face status. The result is summarized in Figure E16(b) as a table. The inter-group and inter-MRV relationships within each group are detailed graphically in Figure E16(c). The information model FMRV5 is presented in Figure E17 as an example.

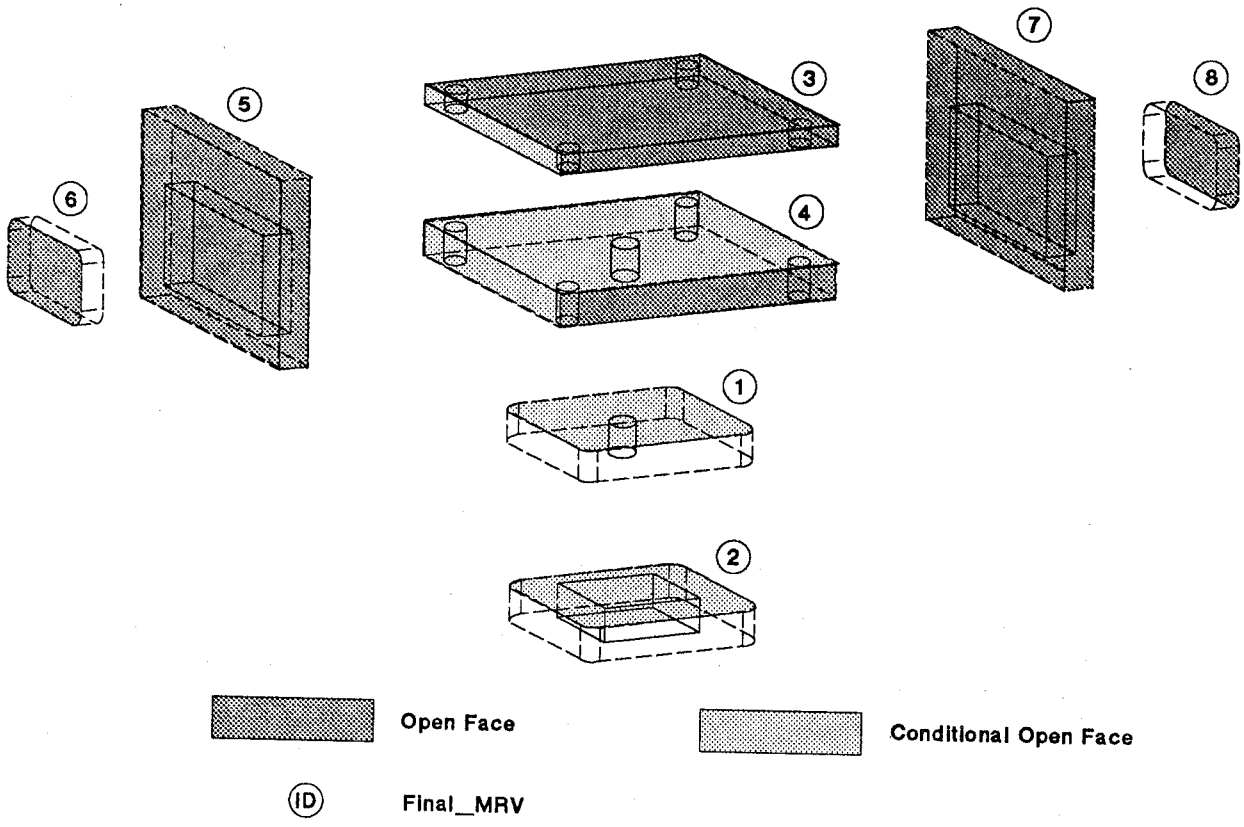
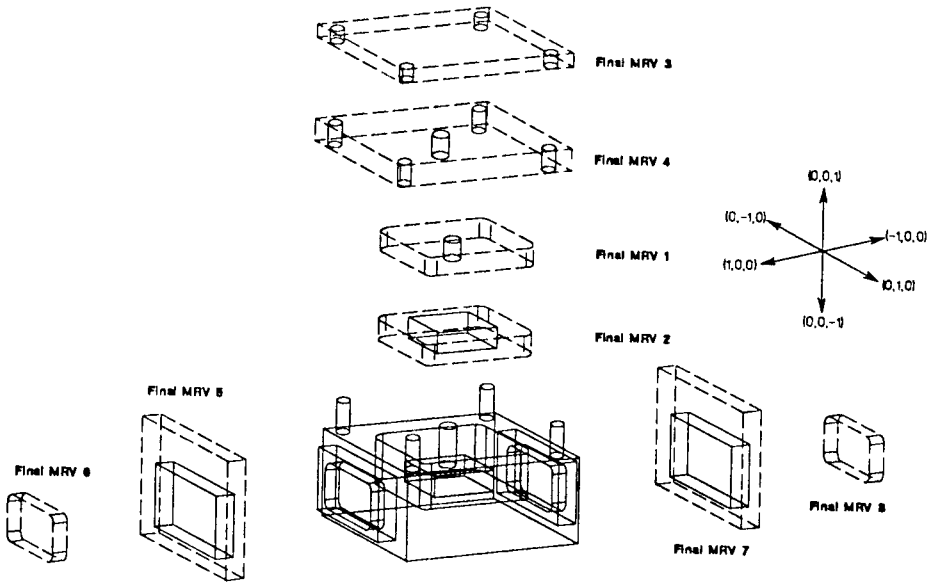


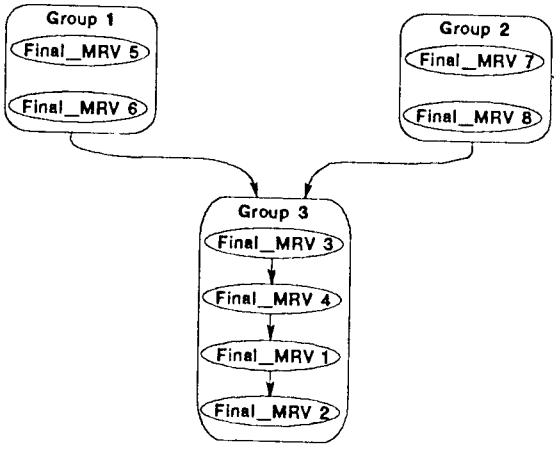
Figure E15: Open Face Status of Final MRVs



(a) Final MRVs

MRV Group ID:	Orientation:	Inter-Group Constraints:	Final MRV ID:	MRV Constraints within Group:
1	(1,0,0)	-	5	-
			6	-
2	(-1,0,0)	-	7	-
			8	-
3	(0,0,1)	G1, G2	3	-
			4	3
			1	4
			2	1

(b) Group and MRV Constraint Summary



(c) Group / Final MRV Precedence

Figure E16: An Illustration of Final MRV Grouping

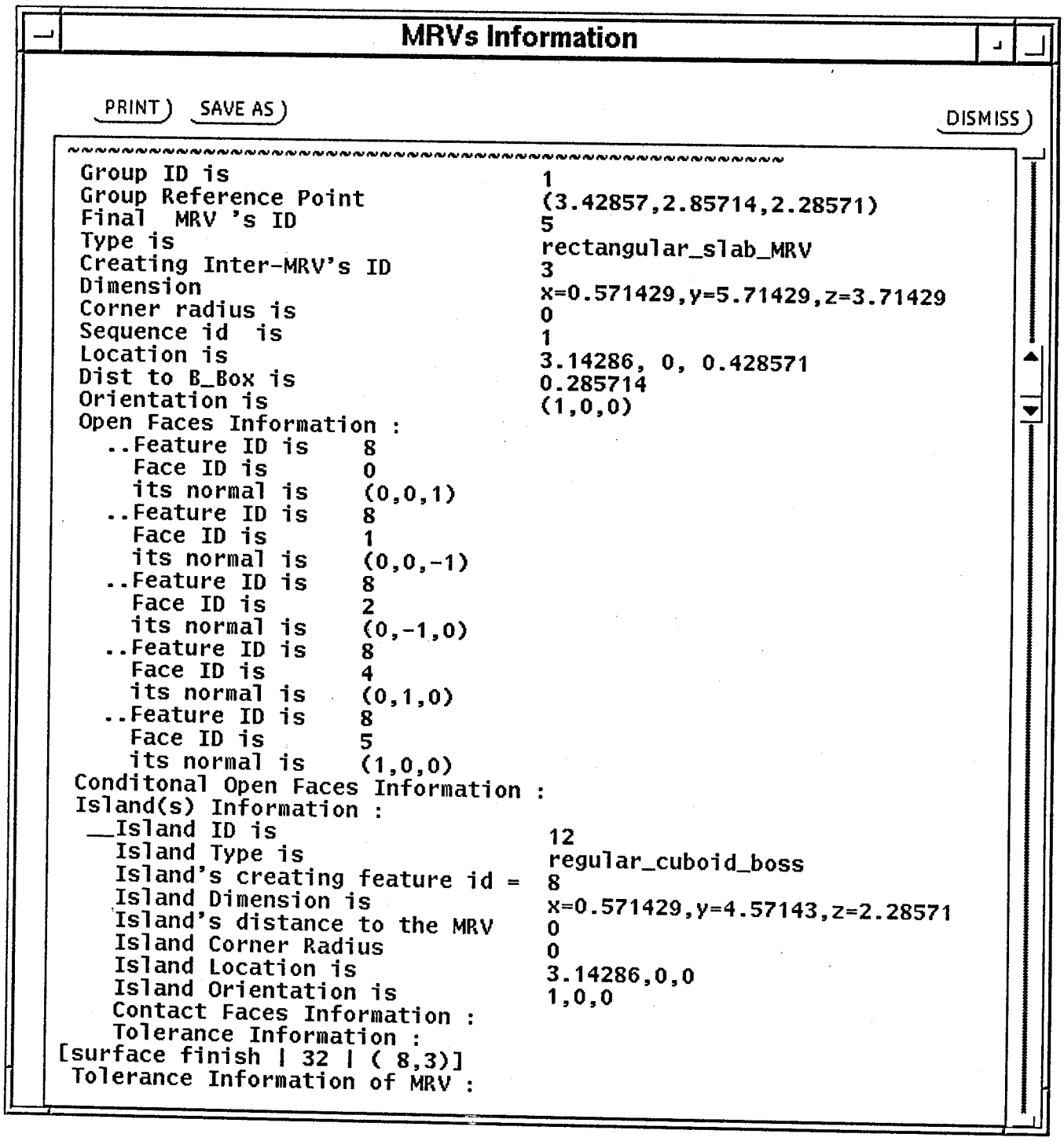


Figure E17: Information Model for a Final MRV

E2.1.3 Stock Selection

The last step in this planning phase is to select a raw material stock. The part material is designated in the product model by specifying the material's name and its AISI/SAE (American Iron and Steel Institute & Society of Automotive Engineers) code. Four stock shapes are modeled for evaluation in this implementation: rectangular, hexagonal, octagonal, and cylindrical. When the search for a stock instance is initiated, stock instances with the specified name and type are examined. The smallest stock that is large enough to accommodate the part design is selected. The search and selection procedure follows a shape order which is pre-arranged to facilitate raw stock fixturing and reduce material waste.

If the selected stock is larger than the bounding box of the part design which is usually the case, the additional material between the selected stock and the part's bounding box is evaluated and converted into appropriate slab MRVs. To do so, the center of the bounding box assumes its location at the stock's center. Cuboid slab MRVs are created by calculating the excessive material volume between the stock's bounding box and the part envelope in each major axis direction. The procedure thus generates six cuboid slab MRVs attached to each major face of the bounding box. Figure E18 illustrates the slab MRVs generated when a cylindrical raw stock is selected. Dotted lines represent the MRVs, while solid lines indicate the actual volumes to be removed from the material stock. In this example, the MRVs are not constrained by any island and the slabs are to be machined in their entirety in order to create the initial cuboid shape of the bounding box before other machining operations are performed on the part design.

E2.2 Operations Planning (Phase 2)

Machine and tool selection for each manufacturing feature (MRV) is the focus of the second planning phase. To properly select machines and tools, two component systems are used: (1) the manufacturing resource database storing the resource instances on which the selection of resources is based, and (2) the manufacturing knowledge base storing human expertise which guides and performs the selection process. The manufacturing resource database has been described elsewhere in this final report. To better understand the machine and tool selection procedure, the manufacturing knowledge structure and its role in resource selection are described here.

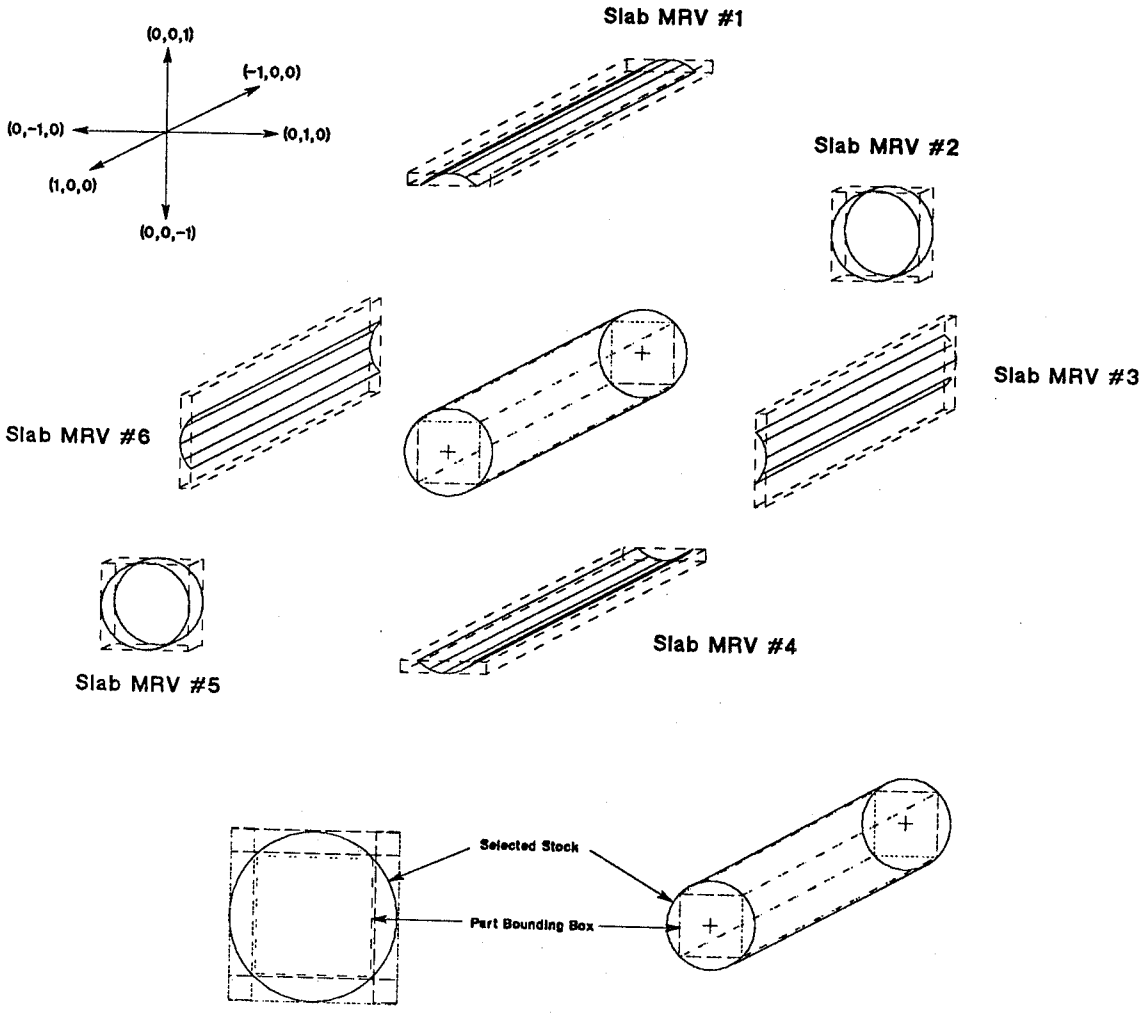


Figure E18: Slab MRVs created from a Cylindrical Stock

E2.2.1 Feature/Machine/Tool (F/M/T) Tables

The capability of machining a form feature is determined jointly by machines and cutting tools. In order to systematically capture the processing capability of a machine shop, its machines and tools have to be first categorized into object classes. Among many attributes which may be used for the classification, their commonality for feature processing needs to be included. In this implementation, machines are grouped into 17 classes and cutting tools are grouped into 27 classes. Each manufacturing feature (defined in terms of MRV) is manually evaluated for each possible machine and tool type combination. The capability of processing each MRV is documented in one F/M/T table and therefore the number of the tables is equal to the number of manufacturing feature types. A new F/M/T table must be manually created for each new form feature type to be added to the system. The size of these F/M/T tables in terms of rows and columns increases when a new machine type or a new type is added to the system. After the initialization stage, these expansions do not occur routinely.

In this implementation, an F/M/T table consists of 27 rows and 17 columns in a 2D matrix. The rows list tool classes while the columns correspond to the machine classes. Each table records all possibly feasible machine and tool class combinations for processing an MRV type. That is, if an MRV can be potentially processed using a particular pair of machine and tool class, the corresponding cell in the table is checked and designated as a feasible combination. A checked combination in an F/M/T table represents only the primary machine and tool classes used to process the MRV. In other words, they are the machine and tool types that will produce the final geometry of a form feature. Figure E19 illustrates the F/M/T table for a manufacturing feature (MRV) called cylindrical_through_hole. A total of 34 possible machine and tool combinations are recorded on the table as feasible for the MRV type. They include:

- 3-axis manual and NC milling machines using endmills, drills, reamers, and boring tools,
- 2-axis manual and NC lathes using single-point turning tools, drills, reamers, and boring tools,
- NC EDM using electrodes,
- NC Wire (EDM) using wire,
- manual and NC jig grinders using grinding wheels,
- bandsaws using a band-type blade,

MRV TYPE: CYLINDRICAL_THRU_HOLE

MRV ID: 68

TOOL ASSEMBLIES			MACHINES																
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	Number		Mill -3-Axis	NcMill-3Axis	Lathe-2Axis	NcLathe-2Axis	EDM	NcEDM	NcEDWC	SurfaceGrind	JigGrind	NcJigGrind	BandSaw	PowerHackSaw	Shape	LightDutyDrill	BedDrill	NcBedDrill	RadialDrill
Mill	1	Face																	
	2	Side																	
	3	End	X	X															
	4	Peripheral																	
	5	MCenter																	
	6	MDrill	X	X															
	7	MReam	X	X															
	8	MBore	X	X															
Lathe	9	ExternalSP																	
	10	InternalSP			X	X													
	11	Form																	
	12	Knurl																	
	13	Groove																	
	14	LCenter																	
	15	LDrill			X	X													
	16	LReam			X	X													
	17	LBore			X	X													
Thermal	18	Electrode					X	X											
	19	Wire							X										
Grind	20	Grind									X	X							
Saw	21	Band																	
	22	Blade											X						
Shape	23	Shape																	
Drill	24	DCenter																	
	25	DDrill																	
	26	DReam														X	X	X	X
	27	DBore														X	X	X	X
																X	X	X	X

Figure E19: MRV/M/T Table for Cylindrical Through Hole MRV

- light duty, manual bed-type, NC bed-type, and radial drilling machines using drills, reamers and boring tools.

E 2.2.2 Manufacturing Knowledge Modules

The MRV/M/T tables are the starting point for the selection of machines and tools. In reality, each check mark in the tables is a pointer pointing to a manufacturing knowledge module which evaluates the feasibility of machine and tool instances through the use of rules. Knowledge rules are categorized within each module to evaluate the feasibility of machine and tool instances in the class for a given manufacturing feature.

As structured in Figure E20, a manufacturing knowledge module consists of two sets of rules: the initial rule set and the main rule set. The initial rule set store pointers to other rule sets, while the main rule set contains rules for the actual feasibility evaluation. Since some pilot operations may be required to properly process an MRV, a knowledge module (especially when it is accessed from an F/M/T table) may point to other knowledge modules from its initial rule set in order to search for the required pilot tools.

Each main rule set captures rules for machine and tool feasibility evaluation at instance level. Machine related rules check machine feasibility in three areas: machine work envelope, tool travel, and weight capacity checking. Tool related rules evaluate tool feasibility in six areas: material cutting capability, cutter body dimensions, overall tool assembly dimensions, island restriction, tool interference, and finally parameter updates. The parameter update rules are applied, only when a feasible tool instance is selected, to update the rule parameters passed on as an input to other knowledge modules. These parameters are used to set a search interval for selecting a tool size for the next knowledge module to be activated. The inputs to a knowledge module include: (1) dimensions of the selected stock, (2) the work envelope of the candidate machine instance, (3) tool travel, (4) weight capacity, (4) tool holder name and type, (5) rule parameters passed from a previous module, (6) the part's AISI/SAE code, and (7) the MRV's information. The output from a knowledge module includes: (1) the ID of the selected machine instance, (2) the selected tool assembly instance ID, (3) an operation instance, and (4) its corresponding elemental MRV. An elemental MRV is the actual material to be removed in an operation using one cutting tool.

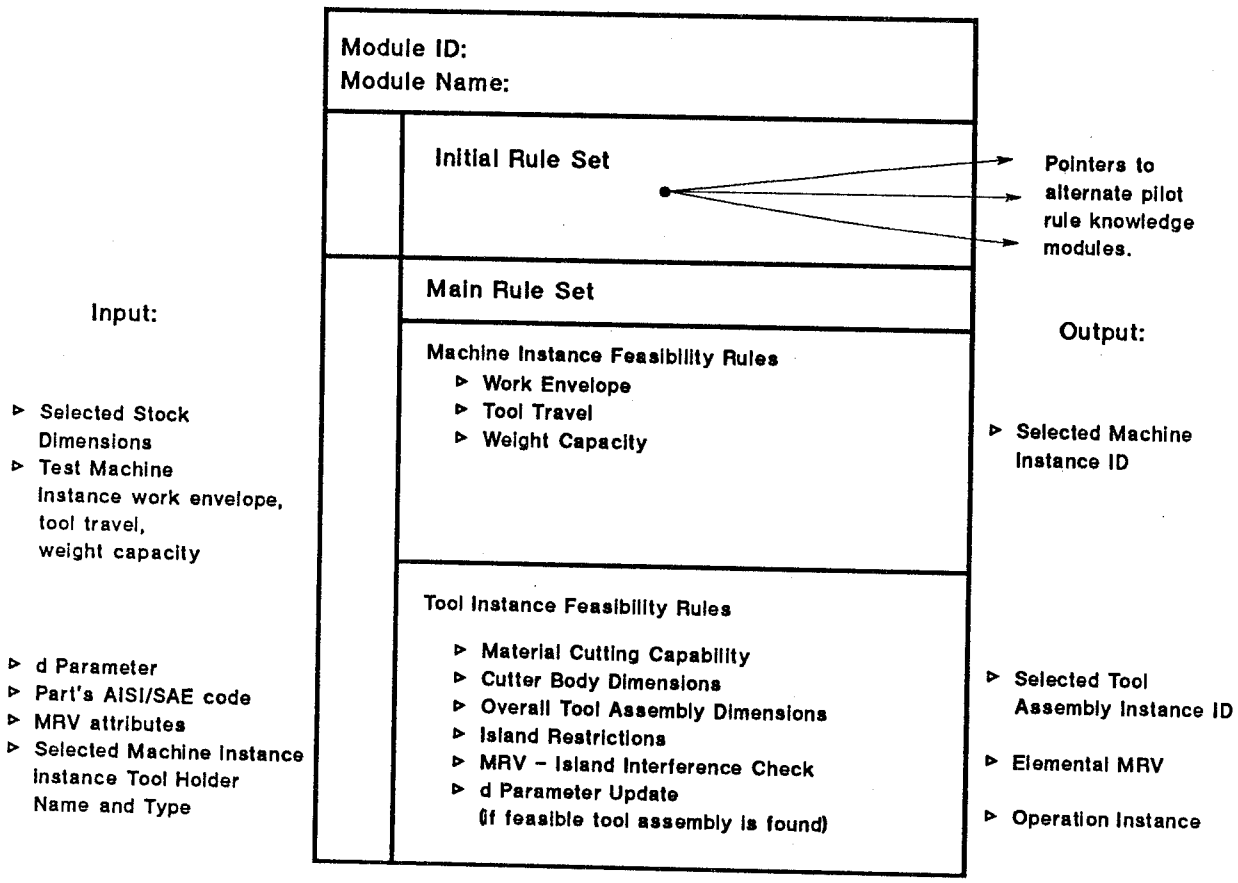
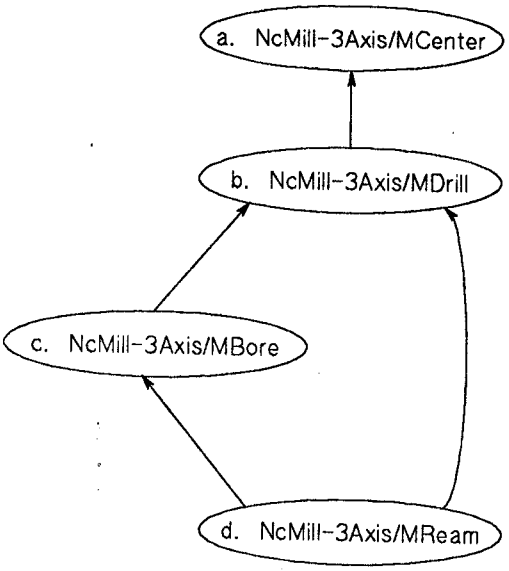


Figure E20: Knowledge Module Structure

The sequence in which different rule modules are called is depicted by the rules in the manufacturing knowledge base. The system evaluates each possible machine and tool combinations for each MRV as specified in the F/M/T tables and according to the knowledge modules which are activated in the evaluation process. Figure E21 illustrate an example application of how knowledge modules interact with one another. In this example, a manufacturing form feature called cylindrical_through_hole is being planned. When the hole is evaluated for the combination of milling machine with reamer, two possible machining sequences are generated. The first operation sequence specifies a center drilling operation, pilot hole drilling operations, boring operation, and final the reaming operation. The other sequence spells the need for a center drilling, pilot hole drilling, and then the reaming operation. In the first operation sequence, the purpose of center drilling is to create a tapered surface so that the subsequent hole to be drilled can be started with ease and no slippage. Pilot hole drilling creates a preliminary hole such that the boring tool can properly remove material. Boring is used to enlarge the hole such that the reamer can produce the hole with the required precision and surface finish. It is possible to ream the hole without the boring operation. It occurs when the initial rules of the NcMill-3axis/Mream knowledge module points to the NcMill-3axis/MDrill as shown in Figure E21.

The process of generating the first operation sequence is shown in Figure E22. When the knowledge module of NcMill- 3axis/Mream is initiated for the cylindrical_through_hole MRV, all machine instances in the machine class *NcMill-3Axis* are evaluated with the machine feasibility rules in the main rule set. These machine rules are applied to only unevaluated machine instances once. If a feasible machine is found, it is temporarily stored and an appropriate tool instance of the tool class *Mream* is searched for and evaluated. If a feasible reamer is found compatible with the stored machine instance, an operation instance is created and temporarily stored together with its corresponding elemental MRV; and the rule parameters are updated accordingly. The module's initial rule set is then evaluated to identify the need for activating other knowledge modules as required for preliminary processing.

As shown in the figure, the NcMill-3Axis and Mream combination was first selected from the MRV table for the cylindrical through hole feature. Its initial rule set identifies the need for preliminary processing and therefore activated the *NcMill-3Axis/MBore* knowledge module for



Feasible Machining Sequences:
A. a, b, c, d **B. a, b, d**

Figure E21: A Knowledge Module Interaction example for a Cylindrical Through Hole MRV

MRV: Cylindrical_Thru_Hole

- Combo: NcMill-3Axis/MReam
- Rule Module Activated: Cylindrical_Thru_Hole MRV / NcMill-3Axis / MReam

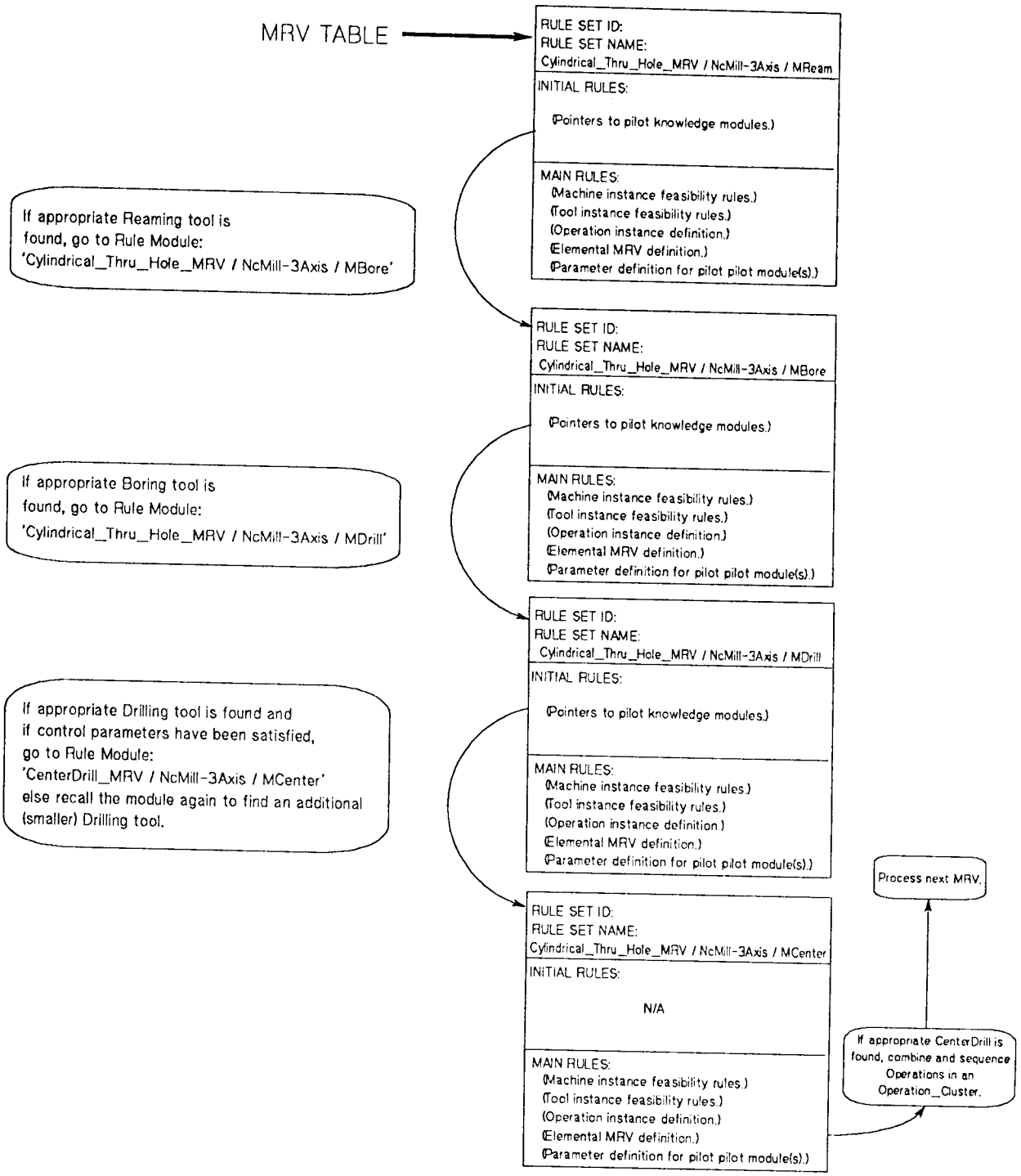


Figure E22: A Knowledge Module Evaluation Example

the same manufacturing feature. A search for an MBore tool in this tool class which is compatible with the selected milling machine instance was launched. After a feasible tool was found, the need for further preliminary processing was again identified and the next knowledge module was activated to satisfy the need. In this example, the Mdrill tool class was called and eventually the need for activating the Mcenter tool class was also identified. After a feasible tool instance was found from each activated knowledge module and the Mcenter does not require any preliminary processing, the search for a feasible machine and tool instance combination is successfully completed. After completing the search, an operation is created for each feasible tool instance identified in the process and temporarily stored in the system with its associated elemental MRV. In the event that the stored machine instance is ultimately selected, these temporarily stored operation instances will be included in the process plan with the selected tools and elemental MRVs. If no feasible tool can be identified in any activated tool class, the search process has to be back tracked.

E2.2.3 Selection Procedure

The procedure for machine and tool selection is shown in Figures E23a through E23e. At the top of Figure E23a, each MRV group is sequentially evaluated. Two MRV lists are initialized for each MRV group. The first list, called the *Processed MRVs* list, contains the IDs of the MRVs for which the required machine and tool instances have been assigned. The other, called the *Remaining MRVs* list, contains the IDs of MRVs which have not been assigned with machine and tool instances. An MRV remains in the list because (1) it has not been evaluated, (2) it has constraining MRV(s) which has not be processed, or (3) it does not have a feasible machine and tool instances. Initially all MRVs are kept in the *Remaining MRV* list.

For every MRV in the *Remaining MRV* list, each of its feasible machine classes is identified from the F/M/T table and attached with a *Processable MRVs* list. Each list contains all MRVs in the *Remaining MRVs* list which can be potentially processed by the machine class. The MRVs in each list are ranked according to their constraining conditions. That is, unconstrained MRVs are placed first in the list, followed by constrained MRVs. The number of MRVs in each *Processable MRVs* list is used to prioritize the search of their corresponding

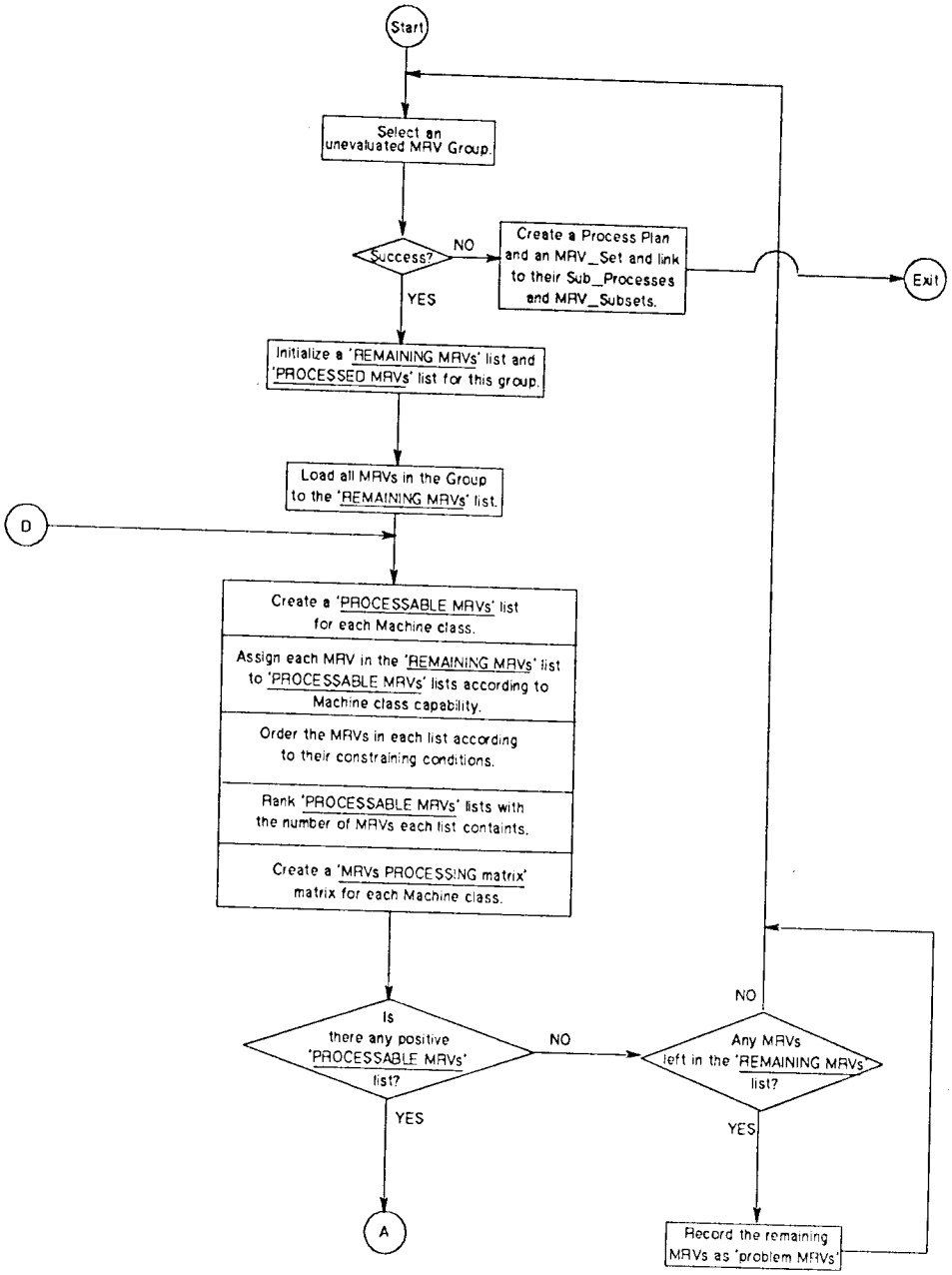


Figure E23a: Machine and Tool Selection Procedure

machine classes. As shown in Figure E23b, the approach is to start the selection procedure by searching first the machine class which can process the largest number of MRVs in order to minimize the number of machine setups.

In order to keep tracks of findings regarding feasible machine and tool instances for the MRVs in the *Processable MRVs* list under evaluation, an *MRVs Processing* matrix is created for each machine class. The rows in the matrix list machine instances in the class, while the columns list all MRVs in the *Processable MRVs* list. When a feasible machine and its compatible tools are found for an MRV, the tool IDs are stored in an appropriate cell in the matrix. The process continues to examine each machine instance and to identify compatible tools required for each MRV. If a feasible machine can not be matched with all required tools, the machine is considered infeasible for the MRV and thus the corresponding cell is marked infeasible. As a result, the machine instance also becomes infeasible for all subsequent MRVs constrained by this MRV.

For the selected machine class, a tool class compatible with both the MRV and the machine class is randomly selected. As shown in Figure E23c, the tool class is evaluated for its capability of meeting the stringiest tolerance and surface finish MRV's specifications. The knowledge module for the machine and the qualified tool class is then activated to identified feasible machine and tool instances for this MRV. As shown in Figure E23d, the selection procedure begins by evaluating a machine instance with the machine feasibility rules in the main rule set of the knowledge module. A feasible is then matched to tool instances in the selected tool class with the tool feasibility rules in the main set. After a feasible tool is found, the initial rule set in the module is activated to evaluate the need for preliminary processing of the MRV. Based on the identified need, an appropriate knowledge module is triggered. After all required tools are identified for processing the MRV on the machine instance, a tentative cluster instance is created and stored in the *MRVs processing* matrix to define all operation instances, their tools, and elemental MRVs.

As depicted in Figure E23e, after the matrix for a machine class is evaluated, the machine instance which can process the greatness number of the MRVs is selected as the challenger to compare to the incumbent machine instance. It replaces the incumbent if the challenger can process more MRVs. The incumbent is compared to the next highest ranked *Processable MRVs*

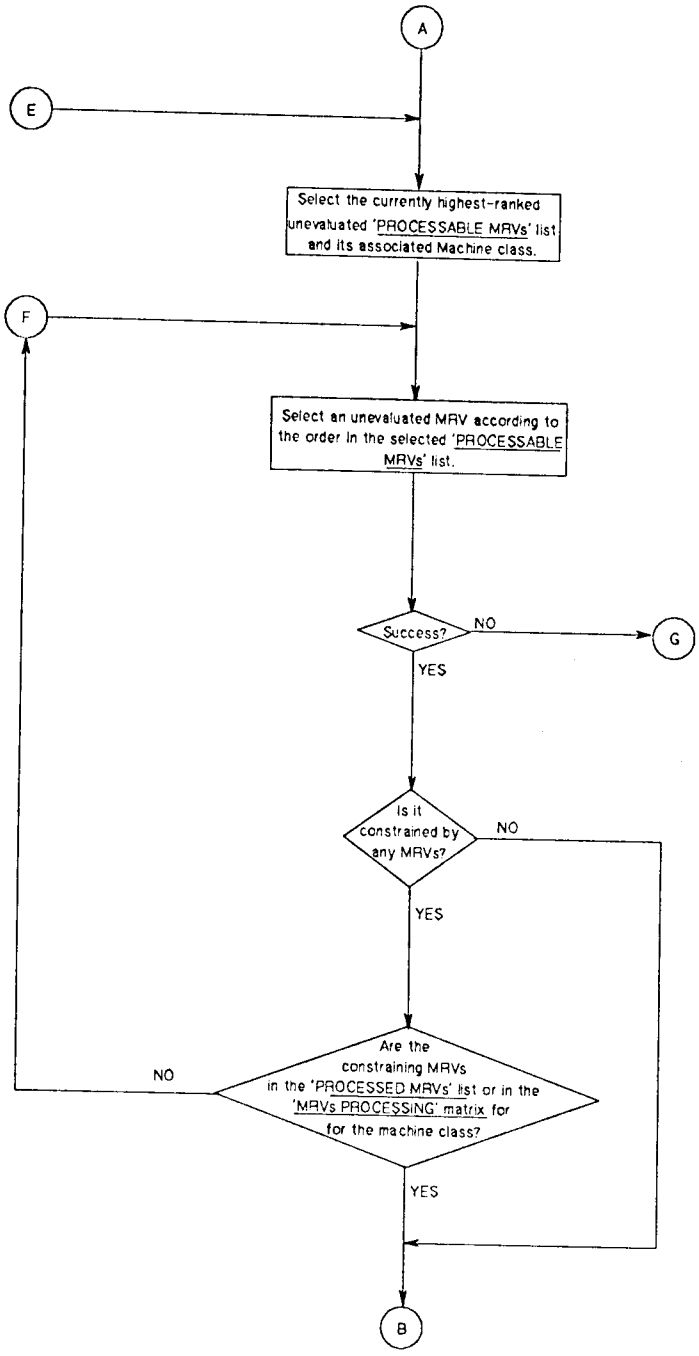


Figure E23b: Machine and Tool Selection Procedure (continued)

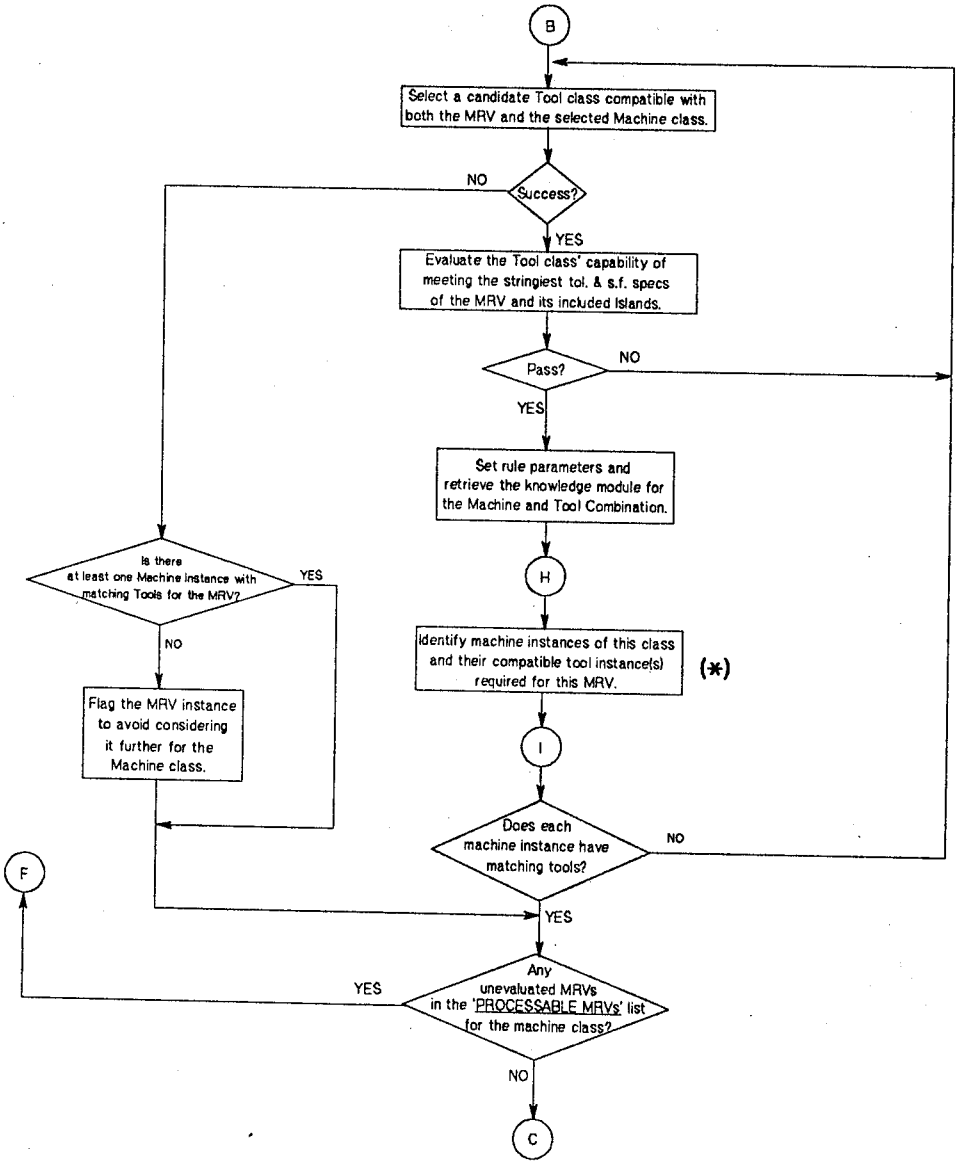


Figure E23c: Machine and Tool Selection Procedure (continued)

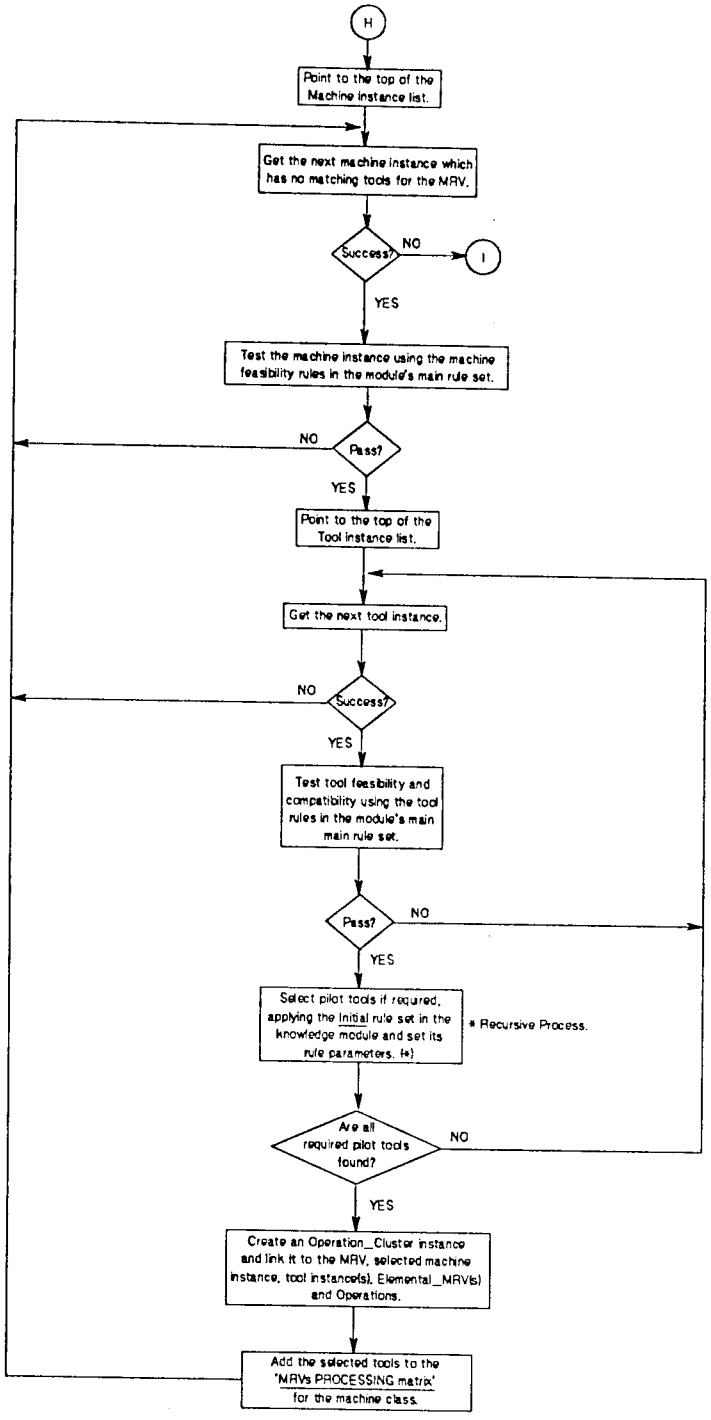


Figure E23d: Machine and Tool Selection (*) (continued)

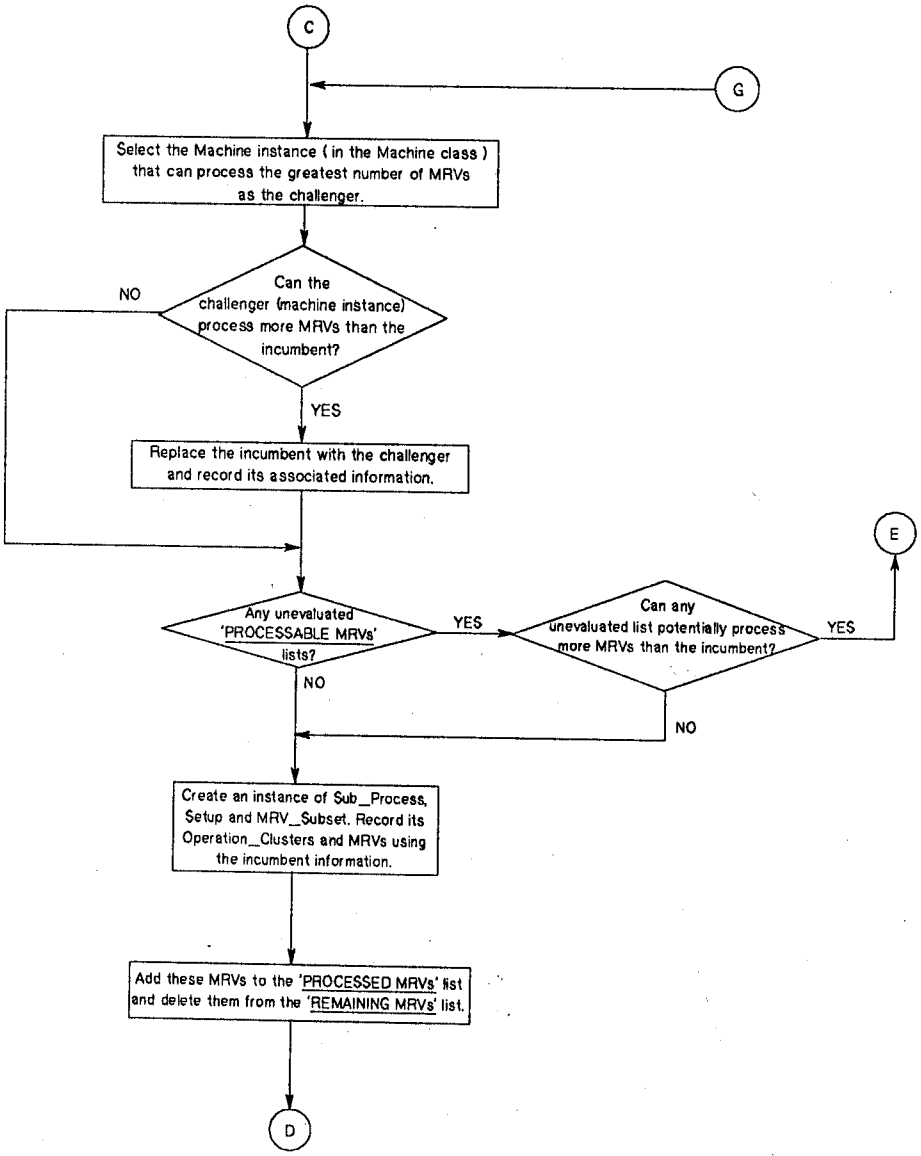


Figure 23e: Machine and Tool Selection Procedure (continued)

list. If the list contains more MRVs than the incumber can process, another *MRVs Processing* matrix is created for its corresponding machine class and the search process resumes for the MRVs in the newly selected *Processable MRVs* list. If no machine can potentially do better than the incumbent, a subprocess is then created to store the MRVs, machine instance, operation clusters, and other process plan elements which are temporarily stored under the current incumbent. All the MRVs included in this subprocess are transferred from the *Remaining MRVs* list to the *Processed MRVs* list. The process continues until all remaining MRVs in the group are assigned or identified as problem MRVs and all MRV groups are processed.

E2.3 Process Plan Evaluation (Phase 3)

This section describes process plan evaluation and illustrate a process plan example. the evaluation process is explained in the first section. The example is given in the second section.

E2.3.1 Process Plan Evaluation

Each process plan instance is evaluated based on the selected objective (minimum number of setups or minimum relative machining cost). If the evaluation does not show an improvement over the threshold value or if no stopping criterion is met, a new process plan is generated for the part design by creating a different MRV sequence or by selecting different manufacturing resources than those chosen in the previously generated process plans. Stopping criteria implemented are computer run time or number of feasible plans stored. When the entire feasible solution space is searched, the best feasible plan previously created is the optimal and the search is terminated. If no plan exists, the problem is reported and the program exits. After the search is over, the best plan is presented to the user for manual review and final approval. Only those plans approved by an authorized reviewer can be stored in the database system.

E2.3.2 A process plan example

A process plan for the sample part as shown in Figure E3 is included at the end of this appendix. The header section includes administrative information such as plan ID and version, planning and approval dates, part ID and version, material specifications, the selected stock ID, and the total number of machine setups required. This plan consists of nine subproceses. Each

subprocess includes the machine instance ID, a description of the setup (added manually by the user), the NC program ID (added when it is created by the NC program generator) and tool access direction information. Also shown in the figure are the operation clusters required for removal of MRVs attached to each subprocess. The individual operations required to completely remove an MRV are located within each cluster with their corresponding elemental MRVs and tool assembly requirement. A total of 20 operations are specified to process this part design on two milling machines and four tool assemblies. The complete MRV information for the part is also included.

```
Plan ID        : 18                Part ID          : 18
Plan Version   : 1                 Part Name        :
Planning Date  : 10/16/1994        AISI/SAE Code    : 5005
Approval Date  :                   Stock  ID        : CR-2
Approved By    :                   Total  Setups    : 9
```

```
Sub Process ID    : 1
Machine/Setup ID  : 1
Description       :

NC Program        :

MRV Subset ID     : 1
Access Direction  : [ 0,0,-1 ]
```

```
Machine ID         : NMTM-8
Machine Name       : NcMill-3Axis
Machine Type       : Vertical
Fixture Assy ID    :
Fixture Assy Type  :
```

```
>Operation Cluster ID : 1-1
 MRV ID               : 106
 MRV Type             : rectangular_slab_MRV

        >Operation  ID  : 1-1-1
         Name           :
         ToolAssy ID    : FaceMill-1
         ToolAssy Name  : Face Mill
         E-MRV ID       : 1
         E-MRV Type     : rectangular_slab
```

```
Sub Process ID    : 2
Machine/Setup ID  : 2
Description       :

NC Program        :

MRV Subset ID     : 2
Access Direction  : [ -1,0,0 ]
```

```
Machine ID         : NMTM-8
Machine Name       : NcMill-3Axis
Machine Type       : Vertical
Fixture Assy ID    :
Fixture Assy Type  :
```

```
>Operation Cluster ID : 2-1
 MRV ID               : 102
 MRV Type             : rectangular_slab_MRV

        >Operation  ID  : 2-1-1
         Name           :
         ToolAssy ID    : FaceMill-1
         ToolAssy Name  : Face Mill
```

```
                        E-MRV ID        : 2
                        E-MRV Type      : rectangular_slab

-------------------------------------------------------------------------
*************************************************************************

       Sub Process ID    : 3
       Machine/Setup ID  : 3
       Description       :

       NC Program        :

       MRV Subset ID     : 3
       Access Direction  : [ 0,1,0 ]
-------------------------------------------------------------------------
       Machine ID         : NMTM-8
       Machine Name       : NcMill-3Axis
       Machine Type       : Vertical
       Fixture Assy ID    :
       Fixture Assy Type  :

-------------------------------------------------------------------------
              >Operation Cluster ID : 3-1
               MRV ID               : 103
               MRV Type             : rectangular_slab_MRV

                      >Operation  ID  : 3-1-1
                       Name           :
                       ToolAssy ID    : FaceMill-1
                       ToolAssy Name  : Face Mill
                       E-MRV ID       : 3
                       E-MRV Type     : rectangular_slab

-------------------------------------------------------------------------
*************************************************************************

       Sub Process ID    : 4
       Machine/Setup ID  : 4
       Description       :

       NC Program        :

       MRV Subset ID     : 4
       Access Direction  : [ 0,-1,0 ]
-------------------------------------------------------------------------
       Machine ID         : NMTM-8
       Machine Name       : NcMill-3Axis
       Machine Type       : Vertical
       Fixture Assy ID    :
       Fixture Assy Type  :

-------------------------------------------------------------------------
              >Operation Cluster ID : 4-1
               MRV ID               : 104
               MRV Type             : rectangular_slab_MRV

                      >Operation  ID  : 4-1-1
                       Name           :
                       ToolAssy ID    : FaceMill-1
                       ToolAssy Name  : Face Mill
                       E-MRV ID       : 4
                       E-MRV Type     : rectangular_slab

-------------------------------------------------------------------------
```

```
**************************************************************************

       Sub Process ID   : 5
       Machine/Setup ID : 5
       Description      :

       NC Program       :

       MRV Subset ID    : 5
       Access Direction : [ 0,0,1 ]
--------------------------------------------------------------------------
       Machine ID       : NMTM-8
       Machine Name     : NcMill-3Axis
       Machine Type     : Vertical
       Fixture Assy ID  :
       Fixture Assy Type :

--------------------------------------------------------------------------
              >Operation Cluster ID : 5-1
               MRV ID               : 105
               MRV Type             : rectangular_slab_MRV

                     >Operation  ID  : 5-1-1
                      Name           :
                      ToolAssy ID    : FaceMill-1
                      ToolAssy Name  : Face Mill
                      E-MRV ID       : 5
                      E-MRV Type     : rectangular_slab

--------------------------------------------------------------------------
**************************************************************************

       Sub Process ID   : 6
       Machine/Setup ID : 6
       Description      :

       NC Program       :

       MRV Subset ID    : 6
       Access Direction : [ 1,0,0 ]
--------------------------------------------------------------------------
       Machine ID       : NMTM-8
       Machine Name     : NcMill-3Axis
       Machine Type     : Vertical
       Fixture Assy ID  :
       Fixture Assy Type :

--------------------------------------------------------------------------
              >Operation Cluster ID : 6-1
               MRV ID               : 101
               MRV Type             : rectangular_slab_MRV

                     >Operation  ID  : 6-1-1
                      Name           :
                      ToolAssy ID    : FaceMill-1
                      ToolAssy Name  : Face Mill
                      E-MRV ID       : 6
                      E-MRV Type     : rectangular_slab

--------------------------------------------------------------------------
**************************************************************************

       Sub Process ID   : 7
       Machine/Setup ID : 7
```

```
        Description      :

        NC Program       :

        MRV Subset ID    : 7
        Access Direction : [ 1,0,0 ]
-------------------------------------------------------------------------
        Machine ID        : NMTM-1
        Machine Name      : NcMill-3Axis
        Machine Type      : Vertical
        Fixture Assy ID   :
        Fixture Assy Type :

-------------------------------------------------------------------------
              >Operation Cluster ID : 7-1
               MRV ID               : 5
               MRV Type             : rectangular_slab_MRV

                     >Operation  ID  : 7-1-1
                      Name           :
                      ToolAssy ID    : EndMill-16
                      ToolAssy Name  : End Mill Tool Assembly
                      E-MRV ID       : 7
                      E-MRV Type     : rectangular_slab

-------------------------------------------------------------------------
              >Operation Cluster ID : 7-2
               MRV ID               : 6
               MRV Type             : rounded_cuboid_pocket_MRV

                     >Operation  ID  : 7-2-1
                      Name           :
                      ToolAssy ID    : EndMill-17
                      ToolAssy Name  : End Mill Tool Assembly
                      E-MRV ID       : 8
                      E-MRV Type     : cylindrical_blind_hole

                     >Operation  ID  : 7-2-2
                      Name           :
                      ToolAssy ID    : EndMill-17
                      ToolAssy Name  : End Mill Tool Assembly
                      E-MRV ID       : 9
                      E-MRV Type     : rounded_cuboid_pocket

-------------------------------------------------------------------------
*************************************************************************

       Sub Process ID   : 8
       Machine/Setup ID : 8
       Description      :

       NC Program       :

       MRV Subset ID    : 8
       Access Direction : [ -1,0,0 ]
-------------------------------------------------------------------------
       Machine ID        : NMTM-1
       Machine Name      : NcMill-3Axis
       Machine Type      : Vertical
       Fixture Assy ID   :
       Fixture Assy Type :

-------------------------------------------------------------------------
             >Operation Cluster ID : 8-1
```

```
                MRV ID                : 7
                MRV Type              : rectangular_slab_MRV

                      >Operation  ID  : 8-1-1
                       Name           :
                       ToolAssy ID    : EndMill-16
                       ToolAssy Name  : End Mill Tool Assembly
                       E-MRV ID       : 10
                       E-MRV Type     : rectangular_slab

-------------------------------------------------------------------------
              >Operation Cluster ID : 8-2
               MRV ID               : 8
               MRV Type             : rounded_cuboid_pocket_MRV

                      >Operation  ID  : 8-2-1
                       Name           :
                       ToolAssy ID    : EndMill-17
                       ToolAssy Name  : End Mill Tool Assembly
                       E-MRV ID       : 11
                       E-MRV Type     : cylindrical_blind_hole

                      >Operation  ID  : 8-2-2
                       Name           :
                       ToolAssy ID    : EndMill-17
                       ToolAssy Name  : End Mill Tool Assembly
                       E-MRV ID       : 12
                       E-MRV Type     : rounded_cuboid_pocket

-------------------------------------------------------------------------
*************************************************************************

       Sub Process ID   : 9
       Machine/Setup ID : 9
       Description      :

       NC Program       :

       MRV Subset ID    : 9
       Access Direction : [ 0,0,1 ]
-------------------------------------------------------------------------
       Machine ID       : NMTM-8
       Machine Name     : NcMill-3Axis
       Machine Type     : Vertical
       Fixture Assy ID  :
       Fixture Assy Type :

-------------------------------------------------------------------------
              >Operation Cluster ID : 9-1
               MRV ID               : 3
               MRV Type             : rectangular_slab_MRV

                      >Operation  ID  : 9-1-1
                       Name           :
                       ToolAssy ID    : EndMill-45
                       ToolAssy Name  : End Mill Tool Assembly
                       E-MRV ID       : 13
                       E-MRV Type     : rectangular_slab

-------------------------------------------------------------------------
              >Operation Cluster ID : 9-2
               MRV ID               : 4
               MRV Type             : rectangular_slab_MRV
```

```
                    >Operation  ID  : 9-2-1
                     Name           :
                     ToolAssy ID    : EndMill-1
                     ToolAssy Name  : End Mill Tool Assembly
                     E-MRV ID       : 14
                     E-MRV Type     : rectangular_slab

-------------------------------------------------------------------------
            >Operation Cluster ID : 9-3
             MRV ID               : 1
             MRV Type             : rounded_cuboid_pocket_MRV

                    >Operation  ID  : 9-3-1
                     Name           :
                     ToolAssy ID    : MDRILL-79
                     ToolAssy Name  : Jobber Drill Assembly
                     E-MRV ID       : 15
                     E-MRV Type     : Tube_Elemental_MRV

                    >Operation  ID  : 9-3-2
                     Name           :
                     ToolAssy ID    : EndMill-45
                     ToolAssy Name  : End Mill Tool Assembly
                     E-MRV ID       : 16
                     E-MRV Type     : cylindrical_blind_hole

                    >Operation  ID  : 9-3-3
                     Name           :
                     ToolAssy ID    : EndMill-45
                     ToolAssy Name  : End Mill Tool Assembly
                     E-MRV ID       : 17
                     E-MRV Type     : rounded_cuboid_pocket

-------------------------------------------------------------------------
            >Operation Cluster ID : 9-4
             MRV ID               : 2
             MRV Type             : rounded_cuboid_pocket_MRV

                    >Operation  ID  : 9-4-1
                     Name           :
                     ToolAssy ID    : MDRILL-79
                     ToolAssy Name  : Jobber Drill Assembly
                     E-MRV ID       : 18
                     E-MRV Type     : Tube_Elemental_MRV

                    >Operation  ID  : 9-4-2
                     Name           :
                     ToolAssy ID    : EndMill-45
                     ToolAssy Name  : End Mill Tool Assembly
                     E-MRV ID       : 19
                     E-MRV Type     : cylindrical_blind_hole

                    >Operation  ID  : 9-4-3
                     Name           :
                     ToolAssy ID    : EndMill-45
                     ToolAssy Name  : End Mill Tool Assembly
                     E-MRV ID       : 20
                     E-MRV Type     : rounded_cuboid_pocket

-------------------------------------------------------------------------
-------------------------------------------------------------------------
```

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

```
Group ID is                              0
Group Reference Point                    (5,0.758929,-1.42857)
Final  MRV 's ID                         106
Type is                                  rectangular_slab_MRV
Creating Inter-MRV's ID                  0
Dimension                                x=10,y=1.51786,z=2.51786
Corner radius is                         0
Sequence id  is                          0
Location is                              0, 0, -2.6875
Dist to B_Box is                         0
Orientation is                           (0,0,-1)
Open Faces Information :
Conditonal Open Faces Information :
Island(s) Information :
Tolerance Information of MRV :
```

```
Final  MRV 's ID                         102
Type is                                  rectangular_slab_MRV
Creating Inter-MRV's ID                  0
Dimension                                x=1.57143,y=8.75,z=8.75
Corner radius is                         0
Sequence id  is                          0
Location is                              -4.21429, 0, 0.428571
Dist to B_Box is                         0
Orientation is                           (-1,0,0)
Open Faces Information :
Conditonal Open Faces Information :
Island(s) Information :
Tolerance Information of MRV :
```

```
Final  MRV 's ID                         103
Type is                                  rectangular_slab_MRV
Creating Inter-MRV's ID                  0
Dimension                                x=10,y=1.51786,z=8.75
Corner radius is                         0
Sequence id  is                          0
Location is                              0, 3.61607, 0.428571
Dist to B_Box is                         0
Orientation is                           (0,1,0)
Open Faces Information :
Conditonal Open Faces Information :
Island(s) Information :
Tolerance Information of MRV :
```

```
Final  MRV 's ID                         104
Type is                                  rectangular_slab_MRV
Creating Inter-MRV's ID                  0
Dimension                                x=10,y=1.51786,z=8.75
Corner radius is                         0
Sequence id  is                          0
Location is                              0, -3.61607, 0.428571
Dist to B_Box is                         0
Orientation is                           (0,-1,0)
Open Faces Information :
Conditonal Open Faces Information :
Island(s) Information :
Tolerance Information of MRV :
```

```
Final  MRV 's ID                         105
Type is                                  rectangular_slab_MRV
Creating Inter-MRV's ID                  0
Dimension                                x=10,y=8.75,z=2.51786
Corner radius is                         0
```

```
 Sequence id  is                          0
 Location is                              0, 0, 3.54464
 Dist to B_Box is                         0
 Orientation is                           (0,0,1)
 Open Faces Information :
 Conditonal Open Faces Information :
 Island(s) Information :
 Tolerance Information of MRV :
 ------------------------------------------
 Final  MRV 's ID                         101
 Type is                                  rectangular_slab_MRV
 Creating Inter-MRV's ID                  0
 Dimension                                x=1.57143,y=8.75,z=8.75
 Corner radius is                         0
 Sequence id  is                          0
 Location is                              4.21429, 0, 0.428571
 Dist to B_Box is                         0
 Orientation is                           (1,0,0)
 Open Faces Information :
 Conditonal Open Faces Information :
 Island(s) Information :
 Tolerance Information of MRV :
 ------------------------------------------
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Group ID is                              1
 Group Reference Point                    (3.42857,2.85714,2.28571)
 Final  MRV 's ID                         5
 Type is                                  rectangular_slab_MRV
 Creating Inter-MRV's ID                  3
 Dimension                                x=0.571429,y=5.71429,z=3.71429
 Corner radius is                         0
 Sequence id  is                          1
 Location is                              3.14286, 0, 0.428571
 Dist to B_Box is                         0.285714
 Orientation is                           (1,0,0)
 Open Faces Information :
   ..Feature ID is     8
     Face ID is        0
     its normal is     (0,0,1)
   ..Feature ID is     8
     Face ID is        1
     its normal is     (0,0,-1)
   ..Feature ID is     8
     Face ID is        2
     its normal is     (0,-1,0)
   ..Feature ID is     8
     Face ID is        4
     its normal is     (0,1,0)
   ..Feature ID is     8
     Face ID is        5
     its normal is     (1,0,0)
 Conditonal Open Faces Information :
 Island(s) Information :
  __Island ID is                          12
    Island Type is                        regular_cuboid_boss
    Island's creating feature id =        8
    Island Dimension is                   x=0.571429,y=4.57143,z=2.28571
    Island's distance to the MRV          0
    Island Corner Radius                  0
    Island Location is                    3.14286,0,0
    Island Orientation is                 1,0,0
    Contact Faces Information :
    Tolerance Information :
[surface finish | 32 | ( 8,3)]
```

```
Tolerance Information of MRV :

Final  MRV 's ID                    6
Type is                             rounded_cuboid_pocket_MRV
Creating Inter-MRV's ID             4
Dimension                           x=0.285714,y=4,z=1.71429
Corner radius is                    0.25
Sequence id  is                     2
Location is                         3.28571, 0, 0
Dist to B_Box is                    0.142857
Orientation is                      (1,0,0)
Open Faces Information :
Conditonal Open Faces Information :
  ..Feature ID is    8
    Face ID is       0
    its normal is    (1,0,0)
  Conditioned by MRV  5
Island(s) Information :
Tolerance Information of MRV :

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Group ID is                         2
Group Reference Point               (-3.42857,-2.85714,-1.42857)
Final  MRV 's ID                    7
Type is                             rectangular_slab_MRV
Creating Inter-MRV's ID             5
Dimension                           x=0.571429,y=5.71429,z=3.71429
Corner radius is                    0
Sequence id  is                     1
Location is                         -3.14286, 0, 0.428571
Dist to B_Box is                    0.285714
Orientation is                      (-1,0,0)
Open Faces Information :
  ..Feature ID is    9
    Face ID is       0
    its normal is    (0,0,1)
  ..Feature ID is    9
    Face ID is       1
    its normal is    (0,0,-1)
  ..Feature ID is    9
    Face ID is       2
    its normal is    (0,-1,0)
  ..Feature ID is    9
    Face ID is       3
    its normal is    (-1,0,0)
  ..Feature ID is    9
    Face ID is       4
    its normal is    (0,1,0)
Conditonal Open Faces Information :
Island(s) Information :
 __Island ID is                     13
   Island Type is                   regular_cuboid_boss
   Island's creating feature id =   9
   Island Dimension is              x=0.571429,y=4.57143,z=2.28571
   Island's distance to the MRV     0
   Island Corner Radius             0
   Island Location is               -3.14286,0,0
   Island Orientation is            -1,0,0
   Contact Faces Information :
   Tolerance Information :
Tolerance Information of MRV :

Final  MRV 's ID                    8
Type is                             rounded_cuboid_pocket_MRV
```

```
Creating Inter-MRV's ID          6
Dimension                        x=0.285714,y=4,z=1.71429
Corner radius is                 0.25
Sequence id  is                  2
Location is                      -3.28571, 0, 0
Dist to B_Box is                 0.142857
Orientation is                   (-1,0,0)
Open Faces Information :
Conditonal Open Faces Information :
  ..Feature ID is    9
    Face ID is       0
    its normal is    (-1,0,0)
  Conditioned by MRV  7
Island(s) Information :
Tolerance Information of MRV :
```

~~~~~~

```
Group ID is                      3
Group Reference Point            (2.85714,2.85714,2.28571)
Final  MRV 's ID                 3
Type is                          rectangular_slab_MRV
Creating Inter-MRV's ID          2
Dimension                        x=5.71429,y=5.71429,z=0.571429
Corner radius is                 0
Sequence id  is                  1
Location is                      0, 0, 2
Dist to B_Box is                 0.285714
Orientation is                   (0,0,1)
Open Faces Information :
  ..Feature ID is    3
    Face ID is       0
    its normal is    (0,0,1)
  ..Feature ID is    3
    Face ID is       2
    its normal is    (0,-1,0)
  ..Feature ID is    3
    Face ID is       4
    its normal is    (0,1,0)
Conditonal Open Faces Information :
  ..Feature ID is    3
    Face ID is       3
    its normal is    (-1,0,0)
  Conditioned by MRV  7
  ..Feature ID is    3
    Face ID is       5
    its normal is    (1,0,0)
  Conditioned by MRV  5
Island(s) Information :
 __Island ID is                  3
   Island Type is                cylindrical_boss
   Island's creating feature id = 4
   Island Dimension is           rx=0.285714,ry=0.285714,h=0.571429
   Island's distance to the MRV  0
   Island Corner Radius          0
   Island Location is            2.42857,2.42857,2
   Island Orientation is         0,0,1
   Contact Faces Information :
   Tolerance Information :
 __Island ID is                  4
   Island Type is                cylindrical_boss
   Island's creating feature id = 5
   Island Dimension is           rx=0.285714,ry=0.285714,h=0.571429
   Island's distance to the MRV  0
   Island Corner Radius          0
```
~~~~~~

```
    Island Location is                2.42857,-2.42857,2
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
  __Island ID is                      5
    Island Type is                    cylindrical_boss
    Island's creating feature id =    6
    Island Dimension is               rx=0.285714,ry=0.285714,h=0.571429
    Island's distance to the MRV      0
    Island Corner Radius              0
    Island Location is                -2.42857,-2.42857,2
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
  __Island ID is                      6
    Island Type is                    cylindrical_boss
    Island's creating feature id =    7
    Island Dimension is               rx=0.285714,ry=0.285714,h=0.571429
    Island's distance to the MRV      0
    Island Corner Radius              0
    Island Location is                -2.42857,2.42857,2
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
[surface finish | 32 | ( 7,4)]
[position | 0.001 | B C | ( 7)| MMC]
 Tolerance Information of MRV :

 Final  MRV 's ID                     4
 Type is                              rectangular_slab_MRV
 Creating Inter-MRV's ID              2
 Dimension                            x=5.71429,y=5.71429,z=0.285714
 Corner radius is                     0
 Sequence id  is                      2
 Location is                          0, 0, 1.57143
 Dist to B_Box is                     0.714286
 Orientation is                       (0,0,1)
 Open Faces Information :
   ..Feature ID is     3
     Face ID is        2
     its normal is     (0,-1,0)
   ..Feature ID is     3
     Face ID is        4
     its normal is     (0,1,0)
 Conditonal Open Faces Information :
   ..Feature ID is     3
     Face ID is        3
     its normal is     (-1,0,0)
   Conditioned by MRV  7
   ..Feature ID is     3
     Face ID is        5
     its normal is     (1,0,0)
   Conditioned by MRV  5
   ..Feature ID is     3
     Face ID is        0
     its normal is     (0,0,1)
   Conditioned by MRV  3
 Island(s) Information :
  __Island ID is                      7
    Island Type is                    cylindrical_boss
    Island's creating feature id =    4
    Island Dimension is               rx=0.285714,ry=0.285714,h=0.285714
    Island's distance to the MRV      0
    Island Corner Radius              0
```

```
    Island Location is                2.42857,2.42857,1.57143
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
  __Island ID is                      8
    Island Type is                    cylindrical_boss
    Island's creating feature id =    5
    Island Dimension is               rx=0.285714,ry=0.285714,h=0.285714
    Island's distance to the MRV      0
    Island Corner Radius              0
    Island Location is                2.42857,-2.42857,1.57143
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
  __Island ID is                      9
    Island Type is                    cylindrical_boss
    Island's creating feature id =    6
    Island Dimension is               rx=0.285714,ry=0.285714,h=0.285714
    Island's distance to the MRV      0
    Island Corner Radius              0
    Island Location is                -2.42857,-2.42857,1.57143
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
  __Island ID is                      10
    Island Type is                    cylindrical_boss
    Island's creating feature id =    7
    Island Dimension is               rx=0.285714,ry=0.285714,h=0.285714
    Island's distance to the MRV      0
    Island Corner Radius              0
    Island Location is                -2.42857,2.42857,1.57143
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
[surface finish | 32 | ( 7,4)]
[position | 0.001 | B C | ( 7)| MMC]
  __Island ID is                      11
    Island Type is                    cylindrical_boss
    Island's creating feature id =    3
    Island Dimension is               rx=0.428571,ry=0.428571,h=0.285714
    Island's distance to the MRV      0
    Island Corner Radius              0
    Island Location is                -0.012923,0,1.57143
    Island Orientation is             0,0,1
    Contact Faces Information :
    Tolerance Information :
Tolerance Information of MRV :

Final  MRV 's ID                      1
Type is                               rounded_cuboid_pocket_MRV
Creating Inter-MRV's ID               1
Dimension                             x=4,y=4,z=0.714286
Corner radius is                      0.25
Sequence id  is                       3
Location is                           0, 0, 1.07143
Dist to B_Box is                      1.21429
Orientation is                        (0,0,1)
Open Faces Information :
Conditonal Open Faces Information :
  ..Feature ID is     0
    Face ID is        0
    its normal is     (0,0,1)
  Conditioned by MRV  4
Island(s) Information :
```

```
  __Island ID is                          1
    Island Type is                        cylindrical_boss
    Island's creating feature id =        3
    Island Dimension is                   rx=0.428571,ry=0.428571,h=0.714286
    Island's distance to the MRV          0
    Island Corner Radius                  0
    Island Location is                    -0.012923,0,1.07143
    Island Orientation is                 0,0,1
    Contact Faces Information :
    Tolerance Information :
 Tolerance Information of MRV :
[surface finish | 32 | ( 1,4)]
```

```
 Final  MRV 's ID                         2
 Type is                                  rounded_cuboid_pocket_MRV
 Creating Inter-MRV's ID                  1
 Dimension                                x=4,y=4,z=0.714286
 Corner radius is                         0.25
 Sequence id  is                          4
 Location is                              0, 0, 0.357143
 Dist to B_Box is                         1.92857
 Orientation is                           (0,0,1)
 Open Faces Information :
 Conditonal Open Faces Information :
   ..Feature ID is     0
     Face ID is        0
     its normal is     (0,0,1)
   Conditioned by MRV  1
 Island(s) Information :
  __Island ID is                          2
    Island Type is                        regular_cuboid_boss
    Island's creating feature id =        2
    Island Dimension is                   x=2.28571,y=2.28571,z=0.714286
    Island's distance to the MRV          0
    Island Corner Radius                  0
    Island Location is                    0,0,0.357143
    Island Orientation is                 0,0,1
    Contact Faces Information :
    Tolerance Information :
[position | 0.01 | B C | ( 2)| MMC]
 Tolerance Information of MRV :
[surface finish | 32 | ( 1,4)]
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The final bounding box = (-3.42857,-2.85714,-1.42857)    (3.42857,2.85714,2.28571)


The select stock 's type is Aluminum
The select stock 's shape is Round
The stock's dimension = length=10, diameter_x=8.75, diameter_y=8.75
The select stock 's AISI code is 5005
The select stock 's ID = CR-2
```

Group #	MRV id	MRV name	Sequence #	Constrained By
0	106	rectangular_slab_MRV	0	
0	102	rectangular_slab_MRV	0	
0	103	rectangular_slab_MRV	0	
0	104	rectangular_slab_MRV	0	
0	105	rectangular_slab_MRV	0	
0	101	rectangular_slab_MRV	0	
1	5	rectangular_slab_MRV	1	
1	6	rounded_cuboid_pocket_MRV	2	5

2	7	rectangular_slab_MRV	1	
2	8	rounded_cuboid_pocket_MRV	2	7
3	3	rectangular_slab_MRV	1	7, 5
3	4	rectangular_slab_MRV	2	7, 5, 3
3	1	rounded_cuboid_pocket_MRV	3	4
3	2	rounded_cuboid_pocket_MRV	4	1

References

1. Roller, D., "Design by Features: An Approach to High Level Shape Manipulation," Computer in Industry, Vol.12, 1989, pp.185-191.

2. Shah, J., and Rogers, M., "A Testbed for Rapid Prototyping of Feature Based Applications," in Advances in Feature Based Manufacturing, by J. Shah, M. Mantyla, and D. Nau (editors), Amsterdam: Elsevier Science Publishing, 1994, pp.423-453.

3. Requicha, A., and Chan, S., "Representation of Geometric Features, Tolerance, and Attributes in Solid Modelers based on Constructive Geometry," IEEE Journal of Robotics and Automation, RA-2(3), 1986, pp.156-166.

4. Chen, C., and Wu, J., "A Hybrid CSG/Brep Representation Scheme for Feature Modeling," Proceedings of CSG'94 Set Theoretic Solid Modeling: Techniques and Applications, Winchester, U.K., April, 1994, pp.291-303.

5. Wu, J., and Chen, C., "Object-oriented Implementation of a Feature Modeling System for CAD/CAM Integration," Computer Aided Design, in review.

6. Wu, J., "Object-Oriented Feature Modeling System for CAD/CAM integration", M.S. Thesis, unpublished, Florida International University, Miami, FL., 1994.

Appendix F: NC Program Generator

Appendix F: NC Program Modeling

F1. Introduction

The feature-based NC program modeling system retrieves a feature-based process plan, generates NC codes, and organizes them into an NC program model. The major tasks for this system are: 1) development of the system framework for automatic NC program modeling, 2) development of an algorithm for orienting the MRV and laying out the work station for setup, 3) development of an algorithm for selection of a tool path strategy, 4) development of an algorithm for selecting machining parameters, 5) development of an algorithm for generation of an NC program, and 6) implementation of these developments into a computer software system. These tasks also led to development of an NC programming knowledge base and the an object class structure for NC codes management.

F2. NC Program Modeling Methodology

The overall system framework for automatic NC program modeling is depicted in Figure F1. It starts with a process plan, identifies MRVs required for an NC program, organizes the required programming data from the selected MRV and resources, determines the tool path, generates the NC codes, verifies the path and NC codes, integrates the NC code segments into a final NC program, stores it as an NC program model, and attaches it to an appropriate subprocess of the retrieved process plan.

The process begins by retrieving an approved process plan and identifying each un-evaluated subprocess. Only the subprocess plans which specify an NC machine require an NC program. Those subprocesses which have an NC program already created need not be programmed either. To program each subprocess, the machine/station instance specified for the subprocess and the corresponding controller are retrieved. According to the process plan structure, a subprocess may contain several operation clusters and each cluster may in turn have multiple machining operations. Thus, the NC program modeling system evaluates one operation (i.e., MRV) of one cluster at a time. Each operation signifies a change in cutting tool and creation of a new MRV. Although MRVs may have constraining island(s) attached to them, each

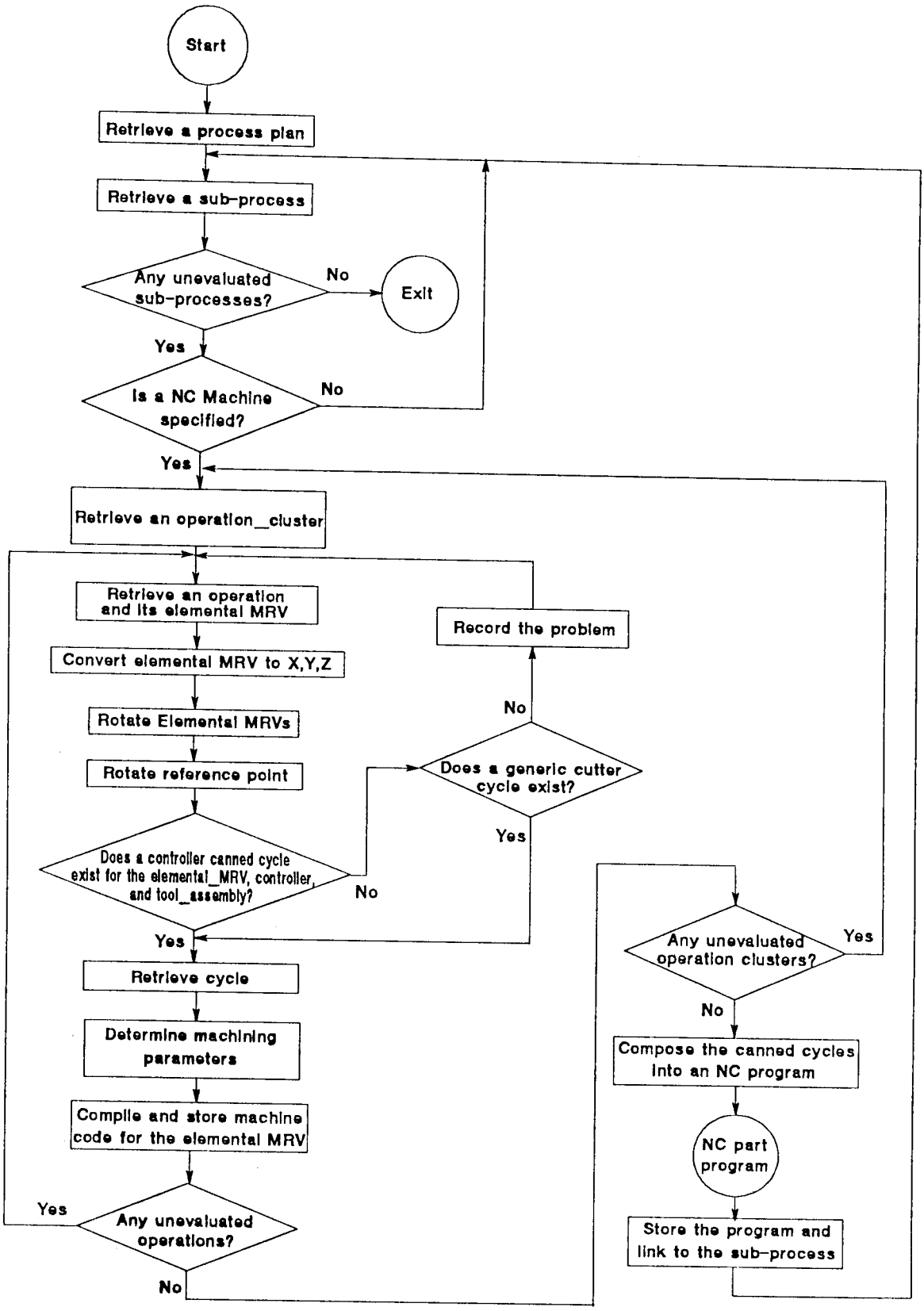


Figure F1: NC Program Modeling Framework

tool selected is presumably feasible for the MRV with/(out) constraining islands.

As an MRV is retrieved and their associated attributes are retrieved, the location, dimensions and orientation of the MRV are re-evaluated to specify the required geometries of the MRV such as extreme points and constraining faces for tool path generation. For each MRV type, a separate algorithm is needed to generate these geometries. The process plan does provide each MRV with the above input data plus a reference point for each MRV with respect to the base part; however, these data must be refined to support the NC programming environment. An algorithm has been developed to change the reference point to zero and reset the vertex points of each MRV to a new value with respect to the reference point in the new orientation. After the rotation and resetting of the reference point related to each MRV is accomplished, the reference point is now set to (0,0,0) for the programming purpose and defined as the starting location in the NC program.

After the preparation is finished, the MRV is evaluated to check if a canned cycle is available on the controller for the selected machine and tool instance. To facilitate the checking procedure, MRVs are categorized into eight groups. Each group represents a major canned cycle type either currently available on a machine controller or having unique characteristics warranted for a separate treatment. All machine controllers, except some older machine controllers, provide some basic prismatic and circular canned cycles. Existing canned cycles on a selected number of machine controllers are identified in Table F1. However, generic canned cycles are critical to ensure the support the wide range of MRV types. The classification and definition of MRVs were based on their open faces and constraining faces. Figure F2 shows the table used to record the mapping relationship between an MRV group and a machine controller. Each cell in the table is a pointer pointing to a pre-conditional rule set to verify the fitness an MRV for existing canned cycles on the controller. Each MRV is evaluated in two manners: with islands and without islands. Figure F3 lists the rules used to evaluate the MRVs in Group E for the BDX32 controller. These rules are compiled to facilitate the search for a canned cycle that performs the correct machining operation based on the given tool, machine controller, elemental MRV, and its constraining island(s).

Figure F4 shows a formatted rule set which consists of tree sections: variable retrieve/calculate, machine parameter retrieve/calculate, and formatted statement. As each MRV

Table F1:
3-Axis Milling Machines
Canned Cycle & Preparatory G-Code Function Relationship

Function	**Anilam Crusader II**	**Bridgeport DX32**	G.E. 1050 MCL	Heidenhain TNC 2500	**Allen Bradley 7320**	Fanuc 11M	Allen Bradley 8200
Rapid Travel	0	0	0	(X)	0	0	0
Linear Travel	1	1	1	(X)	1	1	1
Arc-C.W.	2	2	2	(X)	2	2	2
Arc-C.C.W.	3	3	3	(X)	3	3	3
Ellipse Mill	5						
Cancel Comp	40	40	40	(X)	40	40	40
Cutter Comp L	41	41	41	(X)	41	41	41
Cutter Comp R	42	42	42	(X)	42	42	42
ZigZag Mill		77		(X)			
Pocket Mill	78	78		(X)			
Bore Mill		79					
Cancel Drill	80	80	80	(X)	80	80	80
Drill	81	81	81	(X)	81	81	81
Spot Face	82	82	82	(X)	82	82	82
Deep Drill	83	83	83	(X)	83	83	83
Drill Bore	86	86	85	(X)	86	86	85
Outside Frame		170					
Inside Frame Mill		171					
Pocket Frame Mill	75	172					
Outside Face Mill		173					
Inside Face Mill		174					
Outside Circle		175					
Inside Circle Mill	76	176					
Pocket Circle Mill	77	177		(X)			
Slot Mill		179		(X)			

(X) = Call Functions

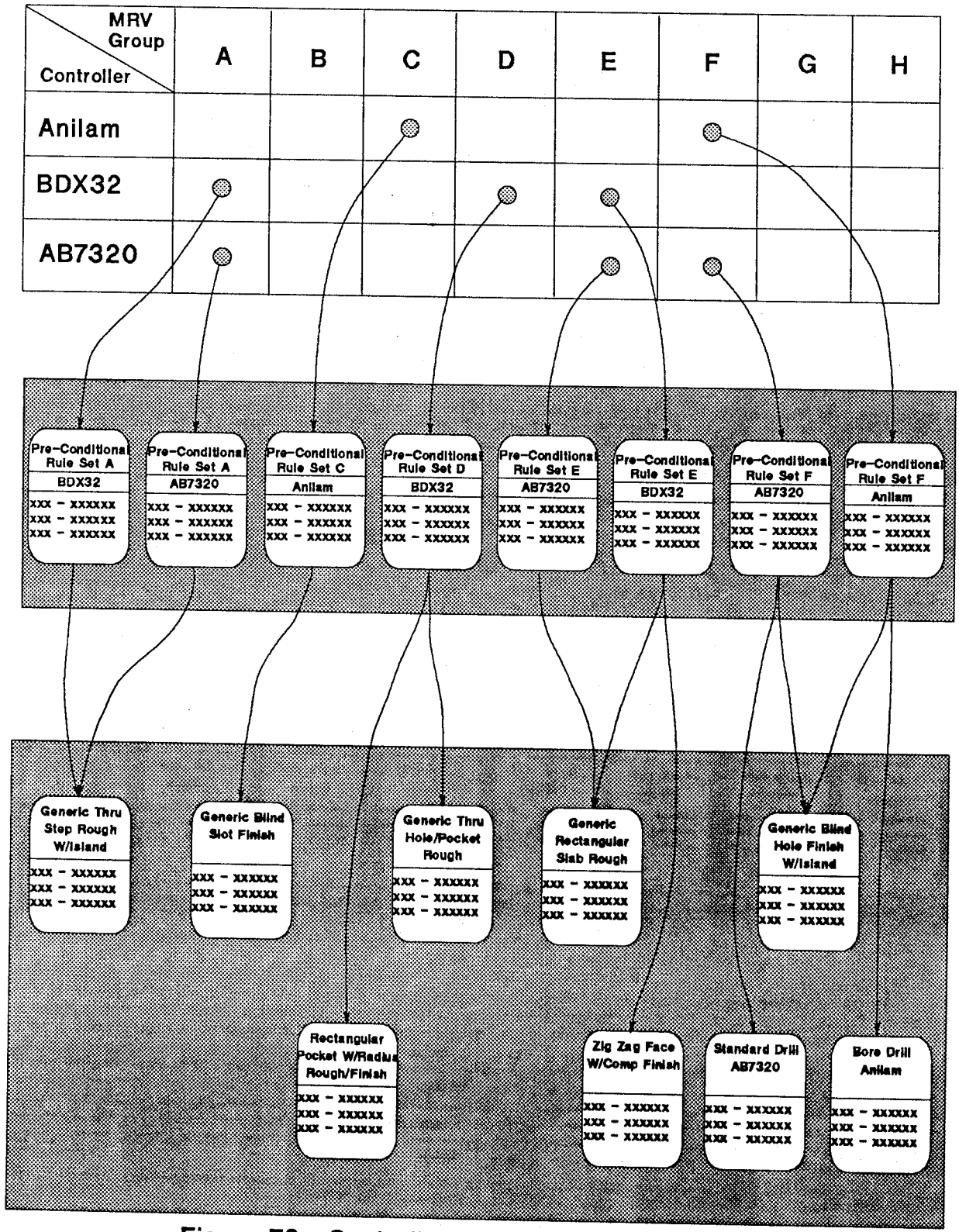


Figure F2 : Controller and MRV Group Mapping

BDX32 - Pre-Conditional Rule Set - Group E

IF Elemental_MRV has an Elemental_Island

- If Elemental_Island = Regular_Cuboid_Boss or Rounded_Cuboid_Boss and Elemental_MRV finish_allowance = 0 Then **BDX32 Frame Finish**
- Else If Elemental_Island = Cylindrical_Boss and Elemental_MRV finish_allowance = 0 Then **BDX32 Circle Finish**
- Else If Elemental_MRV finish_allowance > 0 Then **Generic Rectangular Slab Rough W/Island**
- Else If Elemental_MRV finish_allowance = 0 Then **Generic Rectangular Slab Finish W/Island**
- Else Record Condition

Else IF Elemental_MRV has no Elemental_Island

- IF Elemental_MRV finish_allowance = 0 and Elemental_MRV height (_,_,z) < .25 Then **Zig Zag Face WO/Comp**
- Else IF Elemental_MRV finish_allowance = 0 Then **Zig Zag Face W/Comp Finish**
- Else If Elemental_MRV finish_allowance > 0 Then **Generic Rectangular Slab Rough**
- Else If Elemental_MRV finish_allowance = 0 Then **Generic Rectangular Slab Finish**
- Else Record Condition

Else Record Condition

Figure F3 : Pre-Conditional Rule Set Example

BDX32 - Zig Zag Face W/Comp Finish

VARIABLE RETRIEVE/CALCULATE SECTION

<TOOLNUM> = Set to **Tool Change** value <TOOLNUM>
<XPOS> = (P_{1x} + ((P_{2x} - P_{1x}) / 2))
<YPOS> = (P_{1y} + ((P_{2y} - P_{1y}) / 2))
<XINC> = Elemental MRV dimension (x,_,_): width in X
<YINC> = Elemental MRV dimension (_,y,_): length in Y
<FIPM> = set to ipm
<ZABSRAPID> = Set to .1
<ZDEPTHINC> = set to ZABSRAPID + (Elemental MRV dimension (_,_,z)
<ZDOC> = set to depth_of_cut
<XCLEAR> = (cbdiameter * 1.5)
<OVERLAP> = (cbDiameter * .8)
<FIPMP> = set to ipm * .8

MACHINE PARAMETER RETRIEVE/CALCULATE SECTION

assemblyID = ToolAssembly id
tool_number = Incremental number based on the sequence of Operation
sfm = get from "End Milling" table in variable fpm
rpm = (12 * sfm) / (π * cbDiameter)
ipt = get from "End Milling" table in variable ipt
ipr = ipt * edgeNumber
ipm = rpm * ipr
depth_of_cut = If cbDiameter <= 1.00 then .1
Elseif cbDiameter > 1.00 & <= 3.00 then cbDiameter*.5
else .05
tool_time = n/a

FORMATTED STATEMENT

T<TOOLNUM>//<cbDiameter>; Tool Diameter = <cbDiameter><EOB>
G173X<XPOS>Y<YPOS>Z<ZABSRAPID>X<XINC>Y<YINC>Z<ZDEPTHINC>Z<ZDOC>
P<OVERLAP>P<XCLEAR>F<FIPM>F<FIPMP><EOB>

Figure F4 : Formatted Rule Set Example

is evaluated, its allowance for finish operation is also checked. If an allowance is detected, the NC program modeler will select a roughing canned cycle appropriate to the tool and the MRV. If no allowance is found, a finishing canned cycle is generated, instead. These allowance values are compiled and stored as a table in the knowledge base. As shown in Figure F5, the table is constructed based on cutting tool types and the AISI/SAE codes of material types. Similarly, machining parameters are also compiled and stored in the knowledge base. These parameters with the associated tool information are retrieved after the canned cycle is selected to be stored in the NC program model. Currently the system stores the machining parameters for end milling, drilling, boring, facing milling, and reaming. The table for reaming parameters is exemplified in Figure F6. These tables are managed by the knowledge manager and are accessed from the rule set that is calculating the tool path or producing the canned cycle.

If no canned cycle for the MRV exists on the controller, the generic canned cycle for this MRV group is activated. The generic canned cycle calculates the required tool path according to such machining parameters as the depth of cut, cutting direction, speed, and feed. The tool path is then compiled into the correct machine codes based on the selected controller which are stored in the machine instance. If no generic canned cycle has been developed for this MRV type for this controller, the system will record the problem and continue on to the next MRV. The NC program generation process continues until all MRVs in the cluster and all clusters in the subprocess are finished. The NC segments created for all the MRVs in the subprocess are then compiled into one NC program, stored as an NC program model, and linked to the subprocess. The composition process is shown in Figure F7

F3. NC Program Class Structure

The proposed NC program class structure is shown in Figure F8. Each NC program is defined with five attributes. They are: program ID, process plan ID, sub_process ID, machine ID and controller's ID. Each NC program instance is associated with its corresponding subprocess and can be retrieved through the process plan. Each NC program contains a machine_code object and a machining parameter object. The machining parameters object class contains machining attributes which are stored to support other manufacturing planning applications. In addition to its ID and the tool ID, other attributes defined for this object class

Tool Assembly Class vs Finish Allowance

	Aluminum			Cold	Hot	SS	Titanium
	7075	5005	3003	1018	1020	414	Ti-6Al-4V
Face	0.01	0.01	0.01	0.015	0.02	0.02	0.025
MCenter	0	0	0	0	0	0	0
Side	0.01	0.01	0.01	0.015	0.02	0.02	0.025
MDrill	0	0	0	0	0	0	0
End	0.01	0.01	0.01	0.015	0.02	0.02	0.025
MBore	0	0	0	0	0	0	0
Peripheral	0.01	0.01	0.01	0.015	0.02	0.02	0.025
MReam	0	0	0	0	0	0	0
ExternalSP	0.01	0.01	0.01	0.015	0.02	0.02	0.025
InternalSP	0.01	0.01	0.01	0.015	0.02	0.02	0.025
Groove	0	0	0	0	0	0	0
Form	0	0	0	0	0	0	0
Knurl	0.01	0.01	0.01	0.01	0.01	0.01	0.005
LCenter	0	0	0	0	0	0	0
LDrill	0	0	0	0	0	0	0
LBore	0	0	0	0	0	0	0
LReam	0	0	0	0	0	0	0
Electrode	0.05	0.05	0.05	0.05	0.05	0.05	0.05
Wire	0	0	0	0	0	0	0
Grind	0	0.01	0.01	0.01	0.01	0.01	0.01
Band	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Blade	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Shape	0	0	0	0	0	0	0
DCenter	0	0	0	0	0	0	0
DDrill	0	0	0	0	0	0	0
DReam	0	0	0	0	0	0	0
DBore	0	0	0	0	0	0	0

Figure F5 : Finishing Operation Allowance

Reaming Speeds and Feeds

Material	Hard Low	Hard High	cbMaterial	fpm	ipr (given cbDiameter <=)					
					0.125	0.25	0.5	1	1.5	2
1018	85	124	HSS	45	0.006	0.01	0.015	0.025	0.03	0.035
1018	85	124	Carbide	55	0.006	0.01	0.015	0.025	0.03	0.035
1018	125	174	HSS	40	0.006	0.01	0.015	0.025	0.03	0.035
1018	125	174	Carbide	50	0.006	0.01	0.015	0.025	0.03	0.035
1018	175	224	HSS	35	0.005	0.008	0.012	0.02	0.025	0.03
1018	175	224	Carbide	45	0.005	0.008	0.012	0.02	0.025	0.03
1018	225	275	HSS	30	0.004	0.008	0.012	0.02	0.025	0.03
1018	225	275	Carbide	40	0.004	0.008	0.012	0.02	0.025	0.03
1020	85	124	HSS	45	0.006	0.01	0.015	0.025	0.03	0.035
1020	85	124	Carbide	55	0.006	0.01	0.015	0.025	0.03	0.035
1020	125	174	HSS	40	0.006	0.01	0.015	0.025	0.03	0.035
1020	125	174	Carbide	50	0.006	0.01	0.015	0.025	0.03	0.035
1020	175	224	HSS	35	0.005	0.008	0.012	0.02	0.025	0.03
1020	175	224	Carbide	45	0.005	0.008	0.012	0.02	0.025	0.03
1020	225	275	HSS	30	0.004	0.008	0.012	0.02	0.025	0.03
1020	225	275	Carbide	40	0.004	0.008	0.012	0.02	0.025	0.03
414	225	274	HSS	35	0.004	0.004	0.006	0.008	0.009	0.01
414	225	274	Carbide	50	0.004	0.005	0.006	0.008	0.009	0.01
414	275	325	HSS	35	0.003	0.003	0.004	0.006	0.007	0.008
414	275	325	Carbide	50	0.003	0.004	0.005	0.006	0.007	0.008
414	375	425	HSS	30	0.003	0.003	0.004	0.006	0.007	0.008
414	375	425	Carbide	40	0.003	0.004	0.005	0.006	0.007	0.008
3003	30	80	HSS	180	0.004	0.006	0.01	0.015	0.018	0.02
3003	30	80	Carbide	150	0.004	0.006	0.01	0.015	0.018	0.02
3003	81	150	HSS	180	0.004	0.006	0.01	0.015	0.018	0.02
3003	81	150	Carbide	250	0.004	0.006	0.01	0.015	0.018	0.02
5005	30	80	HSS	180	0.004	0.006	0.01	0.015	0.018	0.02
5005	30	80	Carbide	150	0.004	0.006	0.01	0.015	0.018	0.02
5005	81	150	HSS	180	0.004	0.006	0.01	0.015	0.018	0.02
5005	81	150	Carbide	250	0.004	0.006	0.01	0.015	0.018	0.02
7075	30	80	HSS	180	0.004	0.006	0.01	0.015	0.018	0.02
7075	30	80	Carbide	150	0.004	0.006	0.01	0.015	0.018	0.02
7075	81	150	HSS	180	0.004	0.006	0.01	0.015	0.018	0.02
7075	81	150	Carbide	250	0.004	0.006	0.01	0.015	0.018	0.02
Ti-6Al-4V	310	350	HSS	20	0.003	0.005	0.008	0.01	0.012	0.014
Ti-6Al-4V	310	350	Carbide	35	0.003	0.005	0.008	0.01	0.012	0.014

Figure F6 : Machining Parameters

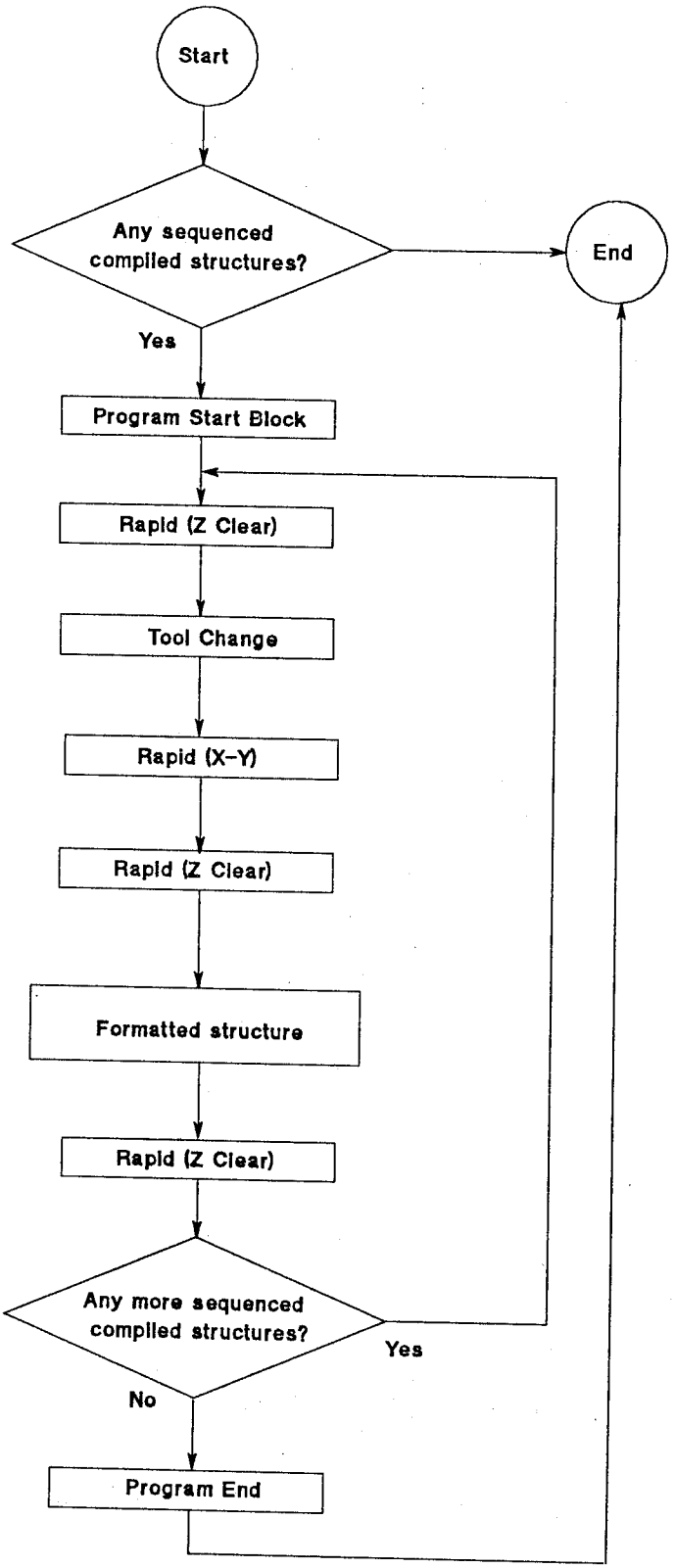


Figure F7 : NC Segments Composition

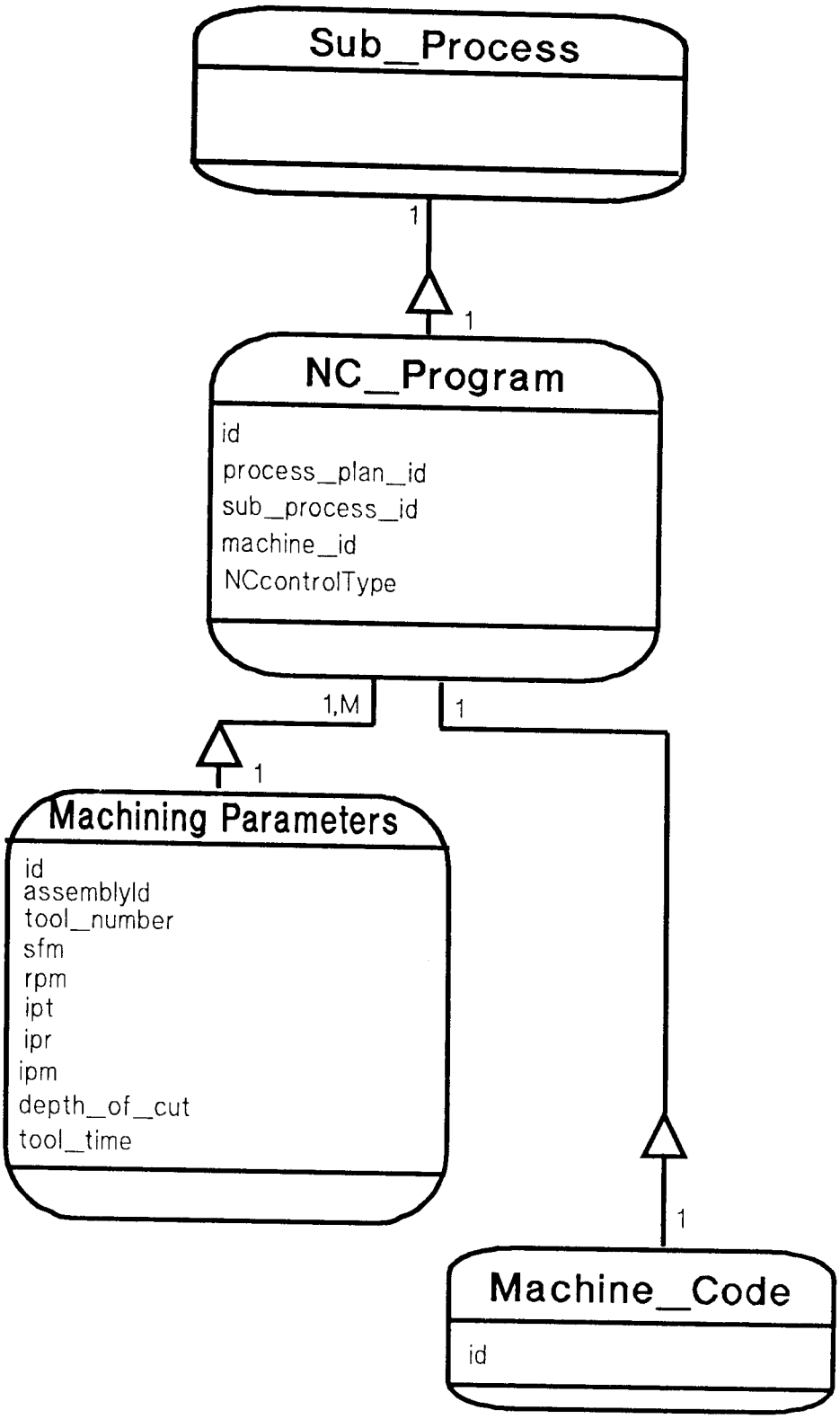


Figure F8 : NC Program Class Structure

include: sfm, rpm, ipt, ipr, ipm, depth_of_cut, and tool_time. These attributes may be used to, for example, set machining parameters on the controller. More frequently they are used to evaluate producibility and estimate manufacturing cost. Attributes such as assembly_id and tool_time can be used with the tool resource manager for tool replenishment evaluation. These attributes are usually instantiated during tool path generation. The machine_code object class has only one attributes denoting the NC program's ID. It is where the actual NC program is stored.

Appendix G: Implementation Summary

1 System Architecture

The environment includes a product modeler, a manufacturing resources manager, a manufacturing knowledge manager, a manufacurablity evaluator, a process planner and a nc program generator. The architecture features a central repository that serves tools both as a manufacturing resources database, knowledge base and a medium for information exchange. The tools in the environment shares this common repository that correlates and incorporates their data(see Figure 1).

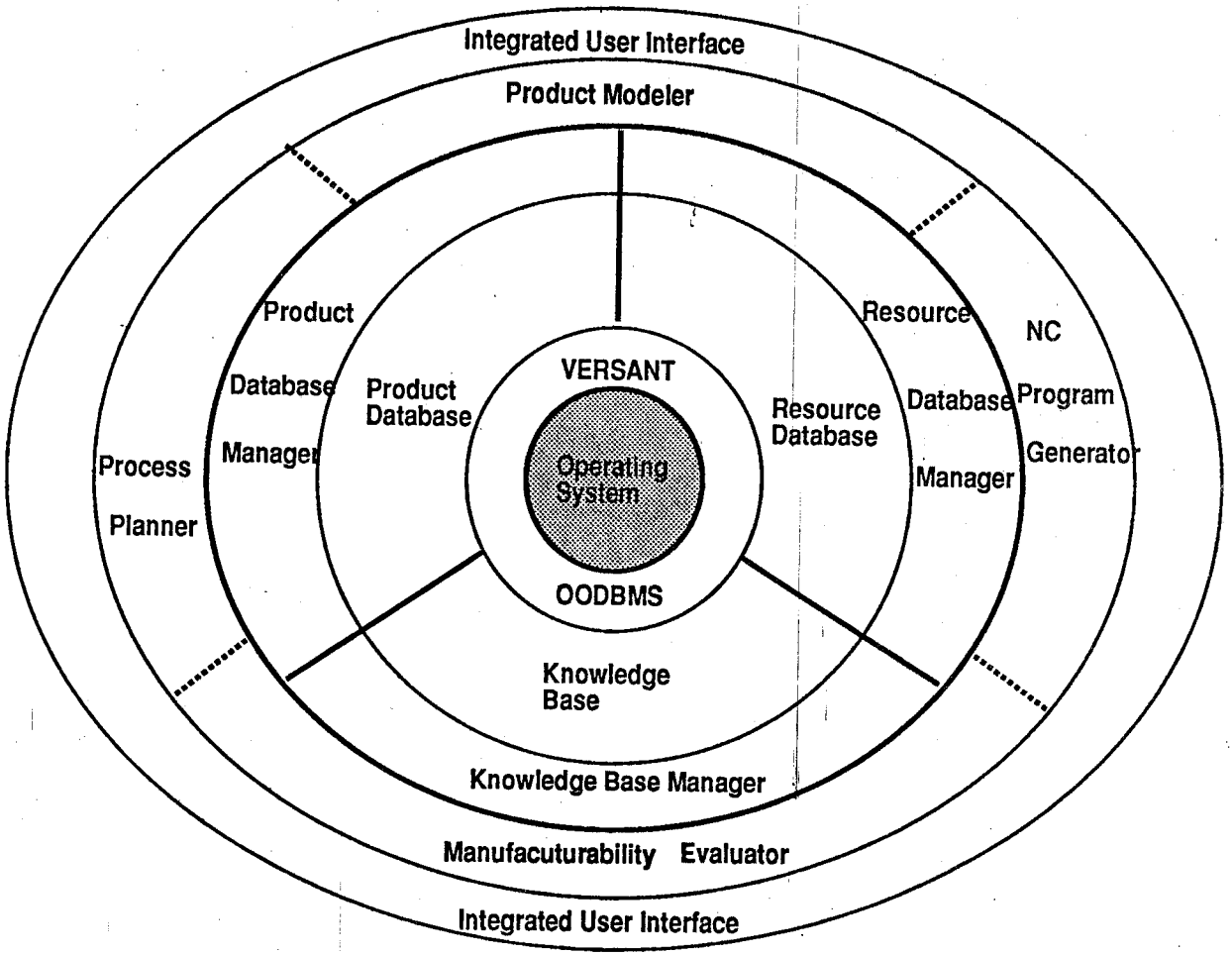


Figure 1: System Architecture Framework

2 The Process Planner

Process planning is the act of preparing detailed processing instructions for manufacturing a part. The process planner will

- convert features into Material Removal Volumes(MRVs),
- sequence the MRVs in the order they are manufactured,
- determine appropriate processes and tools to produce each feature and
- find an optimal machining operation sequence with respect to a suitable criterion.

Figure 2 shows a schematic representation of the components in the process planner, which consist of three basic modules: Converter, Generator and Evaluator. Each of these modules has submodules, each of which performs a specific task. For example, The Generator contains machine selection and tool selection submodules, These are used for selecting suitable machines and tools according to the knowledge stored at machining rule sets. The system consists of many such modules that need to be integrated. Object-oriented design techniques enable self-contained partitioning that leads to a stable model. In other words, all these modules can be encapsulated as individual classes. It makes program understandable and easy to modify to maximize the process planning capability.

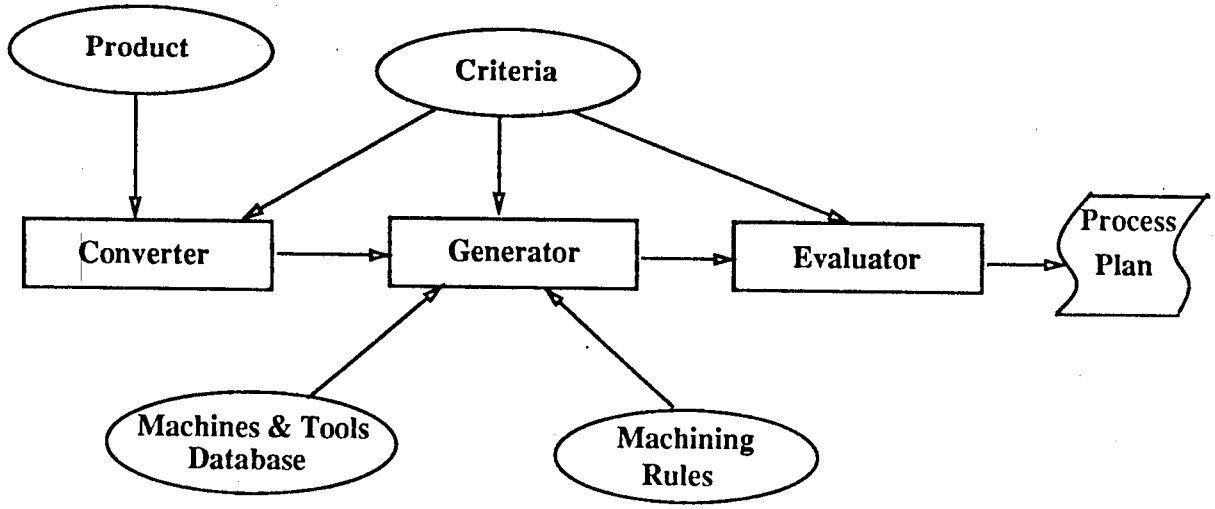


Figure 2: Schematic Representation of The Process Planner

2.1 Class hierarchy

Figure 3 shows the class hierarchy and relationship among them defined in the process planner. In fact, it also includes the class definition required in the NC program generator.

2.2 Converter

The converter accomplishes the conversion of form features into Material Removal Volumes(MRVs) in three steps:

1. Convert features into temporary MRVs and establish their conditional faces,
2. Refine TempMRVs to final MRVs and establish their conditional open faces, and
3. Group final MRVs

2.2.1 Step 1

Initially, form feature are retrieved from the product model in the form of a feature tree. The first feature in the tree is retrieved, which corresponds to the base part, or feature 0. Child depression features are searched for initially. if one is found, an instance of TempMRV (Class for temporary MRV) is created, and specific information of the depression feature is coped to it. Such information includes feature type, dimensions, location, orientation and tolerances. The search is set to this child feature and the query continues to look for other nested depression until all are converted. If none are found, nested child protrusions are searched for. If any are found, affected TempMRVs are constrained, and if the protrusion expands beyond the current bound box, new TempMRV layers added. The search is then set to this features and other nested protrusions are checked. The procedure is repeated until no more protrusions are found. At this point, the search feature is checked if it corresponds to the base part. If there is no match, the search is moved up one level to the search feature's parent. and depression children are again checked for. If the search feature is the base, this means the all depression and protrusion features have been converted/evaluated, and the search ends.

All faces of the TempMRVs have been considered open up to now. An open face is by definition a face though which a tool can enter the MRV in order to remove it. Its open status guarantees there is no material blocking its access. If there is some MRV blocking an open face of another MRV, that open faces becomes conditional open and the removal of the MRV is now dependent upon the prior removal of any blocking MRVs. At first, a TempMRV is retrieved and a candidate open face selected. Then, the MRV is checked if it is slab or a depression-type TempMRV. If the MRV is of the slab MRV, the selected open face is selected any conditioning TempMRVs are

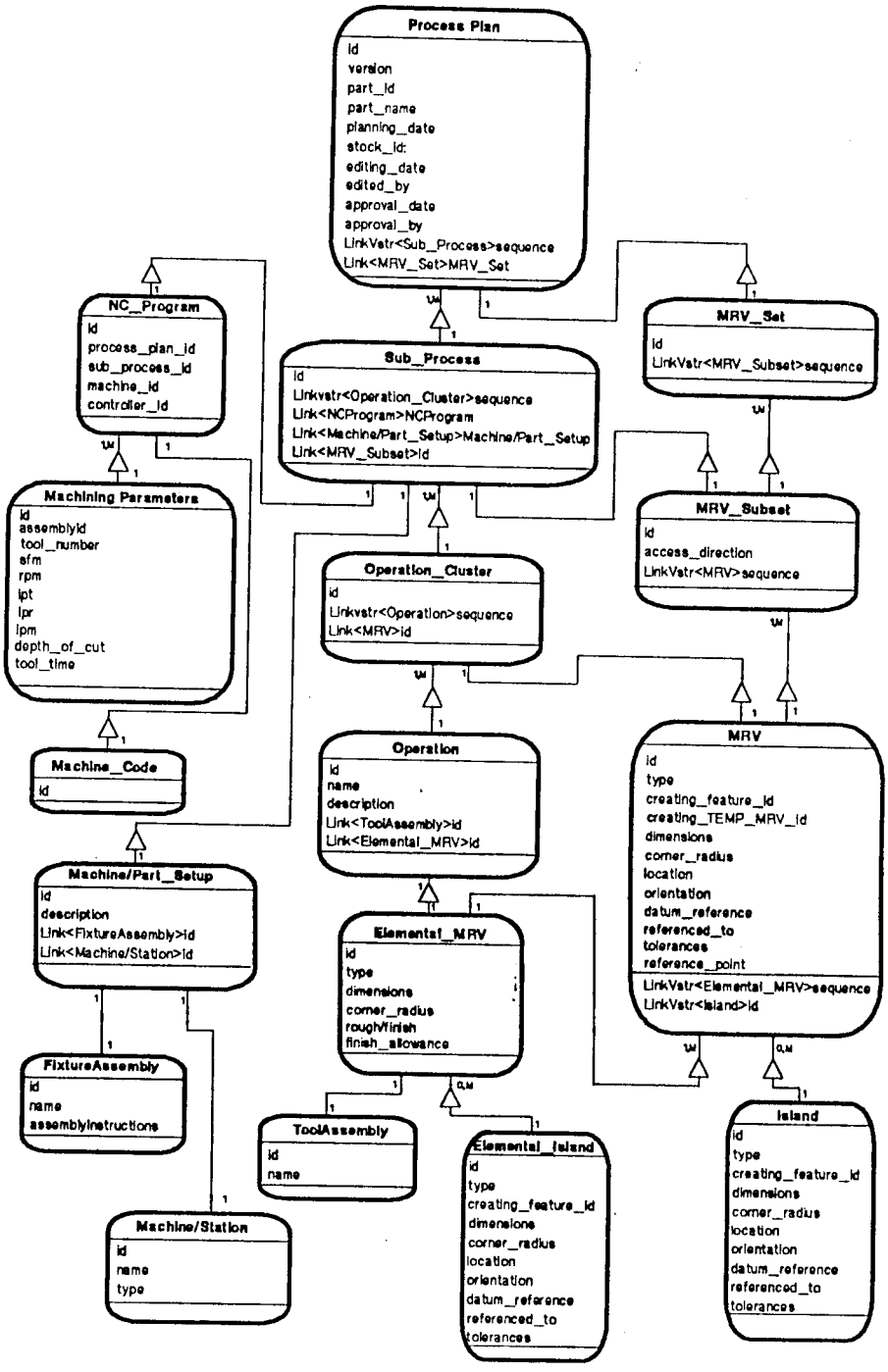


Figure 3: Class Hierarchy on Process Planner

identified. If it is not, the constraining TempMRVs are identified. If constraining TempMRVs are identified, the open face is changed to conditional open and the constraining MRV's ID is recorded. If none were found, other candidate open faces are checked until all have been evaluated. This procedure will be repeated for all TempMRVs.

The major data structures are TempMRV, Bounding-Box. Their definition are following:

```
class TempMRV    : public PObject {
    int                          id;
    PString                      type;
    int                          feature_id;
    double                       cuboid_dim[3];
    double                       prism_dim[3];
    double                       frustrum_dim[4];
    double                       poly_x[100],poly_y[100],poly_h;
    int                          side;
    double                       corner_radius;
    double                       location[3];
    double                       orientation[3];
    Link<FeatureSurface>         f_surface;
    Link<F_Face>                 select_o_face;
    Link<Datum>                  datum_reference;
    Link<Datum>                  reference_to;
    public:
    TempMRV();
   ~TempMRV();

    VVList<F_Face>               contact_faces;
    VVList<F_Face>               constrain_faces;
    VVList<F_Face>               open_faces;
    VVList<C_O_Face>             c_open_faces;
    VVList<Tolerance>            tolerances;
    VVList<TempIsland>           temp_island; };


  Class Bounding_Box  {
        double                       min_p[3];
```

```
double                  max_p[3];
double                  center_point[3];

public:
Bounding_Box();
Bounding_Box(double[],double[],double[]);
~Bounding_Box(); };
```

2.2.2 Step 2

Up to this point, TempMRVs have been fully defined. Such MRVs may have been created to capture the material to be removed inside depression or around protrusions. If there are any protrusions located on a face of the base part or inside a depression TempMRV and extending beyond the base part's original bounding box, slab TempMRVs were created and constrained by TempIslands. In the case of depression-type TempMRVs, they may have associated TempIslands if there were any nested protrusions found inside depression. Multiple TempIslands may be associated to an individual TempMRV. These islands may have different height or may be located on other islands. In order to accurately capture machinable layers, it becomes necessary to refine those TempMRVs having TempIslands into thinner layers, or final MRVs, based on the difference in height between the TempIslands. The open/ conditional open faces of FinalMRVs also need to be specified in order to establish their removal sequence. This performed through a similar procedure used to specify face status for TempMRVs.

The major data structures used in this module are TempIsland and FinalMRV.

```
Class   TempIsland:     public PObject {
        int                     id;
        PString                 type;
        int                     feature_id;
        double                  cuboid_dim[3];
        double                  prism_dim[3];
        double                  frustrum_dim[4];
        double                  poly_x[100],poly_y[100],poly_h;
        int                     side;
```

```
        double              corner_radius;
        double              location[3];
        double              orientation[3];
        double              top2mrv;

public:

        VVList<F_Face>      contact_faces;
        VVList<Tolerance>   tolerance; };


class   FinalMRV        : public PObject {
        int                 id;
        PString             type;
        int                 tempMrvId;
        double              cuboid_dim[3];
        double              prism_dim[3];
        double              frustrum_dim[4];
        double              poly_x[100],poly_y[100],poly_h;
        int                 side;
        double              corner_radius;
        double              location[3];
        double              orientation[3];
        double              dist2bb;
        Link<Datum>         datum_reference;
        Link<Datum>         reference_to;
        int                 machineID;
        int                 toolID;
        int                 groupID;
        int                 sequenceID;
public:
        FinalMRV();
       ~FinalMRV();

        VVList<F_Face>      open_faces;
        VVList<C_O_Face>    c_open_faces;
```

```
    VVList<Tolerance>       tolerances;
    VVList<TempIsland>      islands; };
```

2.2.3 Step 3

In this step, FinalMRVs are grouped based on common orientation. Groups are initially sequenced based on the ID of their creating TempMRV. Then, FinalMRVs within each group are sequenced, based on the status of their conditional open faces. The main data structure:

```
class   MRVGroup : public PObject {
      int                       id;
      double                    refPoint[3];
   public:
      VVList<FinalMRV>          mrvs;

      MRVGroup();
     ~MRVGroup(); };
```

2.3 Generator

The generator produces the candidate process plan based on the knowledge stored int the knowledge base. The work involves machine and tool selection, MRVSet and SubProcess setup.

Initially an unevaluated MRV group is selected. Two lists are then initialized: a PROCESSED MRVS list, which contains the IDs of each MRV for which a machine and required tool assembly instances have been assigned, and a REMAINING MRVS list, which contains the IDs of MRVs that either have not been processed, or have not been assigned machine and tool assembly, or could not be processed because of unevaluated conditioning MRVs or because no feasible machine tools have been found for them. For each machine class a PROCESSABLE MRVS list will be created based the MRV's FMT table. When feasible machine and tool assembly instances are found for an MRV, its ID is copied to the TENTATIVE RPOCESSED MRVS list which is empty initially. An inquiry is then done to find any PROCESSABLE MRVS list with rank greater than 0. If none are found, the REMAINING MRVs list is searched for any left over MRVs. If any are

found, they are recorded as problem MRV either because 1) no feasible machine/tool instances were found for them, 2) they are constrained by other MRVs for which to feasible machine/tool instances were found.

If the selected MRV has conditioning MRVs or its conditioning MRVs have been processed or tentatively processed. a combo which includes the current machine class is chosen, then check the MRV's stringest tolerance against the tool assembly. If it satisfies, The machine/tool selection rule sets linked to the combo are accessed. Rule sets provide the evaluation critiera for feasible machine and tool assembly selection given an MRV type and a combo. Rule sets plays an important role during generating the process plan. All rule sets have following generic framework:

```
INPUT  : MRV's dimension, machine class, tool class.
OUTPUT : selected tool assemblies or error message.

BEGIN

 1. retrieve the all instances of given tool class from the
    database.

 2. filter the instances with machine's material cut
    capability, tool hold type and name.

 3. select a instance with the specific conditions from
    the filtered instances.

4. call other rule set(may be itself ) finish the initial
    operation if necessary.

 5. create the elemental MRVs

 6. return the selected tool assemblies and created
    elemental MRV.

END
```

After feasible machines and their compatible tool instances are identified and recorded together with created elemental MRV, operations, and operation cluster for the selected MRV are created. At last, the whole process plan is compiled based on the operation clusters.

Following are main data structures:

```
class Tool_Op: public PObject {
      PString  description;
      PString  tool_id;
      PString  tool_type;
      PString  emrv_id;
      PString  emrv_type;

      public:

      Link<Element_MRV>  emrv; };


class Tool_List: public PObject {
      o_4b          mrv_id;
      PString       mrv_type;

      public:

      VVList<Tool_Op> tool_ops; };


class Tentative: public PObject {

      PString  mach_id;
      PString  mach_name;
      PString  mach_type;

      public:

      VVList<Tool_List> t_list; };
```

2.4 Evaluator

This module evaluates the candidate process plan generated by the generator and find an optimal one with respect to the criteria: the number of machine setup, cost, time or the combination of these factors. The main data structure used in this module are:

```
class   SubProcess :  public PObject {
        int             id;

        Link<NCProgram>         ncProgram;
        Link<MP_Setup>          mpSetup;
        Link<MRVSubSet>         mrvsset;

        public:
        SubProcess();
       ~SubProcess(); };


class   Plan :  public PObject {
        int             id;
        int             version;
        int             partID;
        PString         partName;
        PString         stockID;
        PString         planingDate;
        PString         approvalDate;
        PString         approvedBy;

        public:
        Plan();
       ~Plan();

        VESet<o_u4b>            problem_mrvs;
        LinkVstr<SubProcess>    sequence;
```

```
Link<MRVSet>            mrvSet; };
```

2.5 User Interface

The generated process plan , an instance of the class Plan , is displayed on an interactive user interface(see Figure 4). In this interface, the user can select a product designed by using Product Modeler, then retrieve the process plans generated before for it, or generate a new one based on the new criterion. The interface provides the utilities for saving process plan into the database or a file, printing a hard copy of the selected process plan and popping up a subwindow to view the MRV information.

3 The NC Program Generator

Once a process plan has been developed, the NC program can take place.the NC program generator will transform the high-level process plan into low level CNC codes based on the selected machine and tool in the plan. Figure 5 shows a schematic representation of the components in the NC program generator.

3.1 User Interface

We have implemented a user interface for the NC program generator, see Figure 6. With this interface, users can retrieve an approved process plan from the database, generate the NC codes for it. then users can edit or modify the generated NC codes to satisfy their special needs. After edition or modification, the NC codes can saved into the database for further use.

4 Manufacturing Database and Its Manager

4.1 Data Structure

The database is constructed on the object-oriented database management system VERSANT. The data in our manufacturing resource database can be classified as:

- material data
- machine data
- tool data

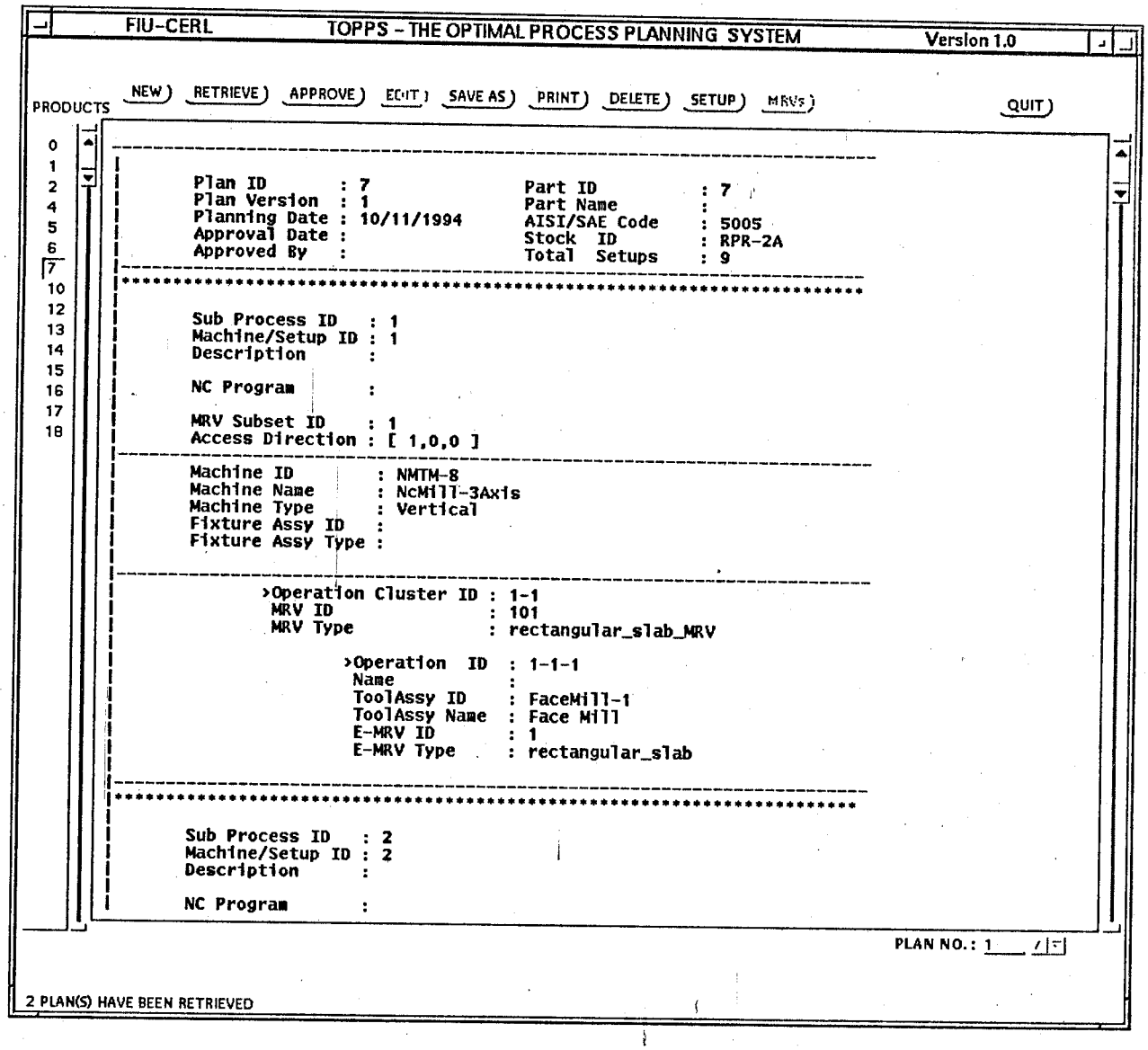


Figure 4: Process Planning Interface

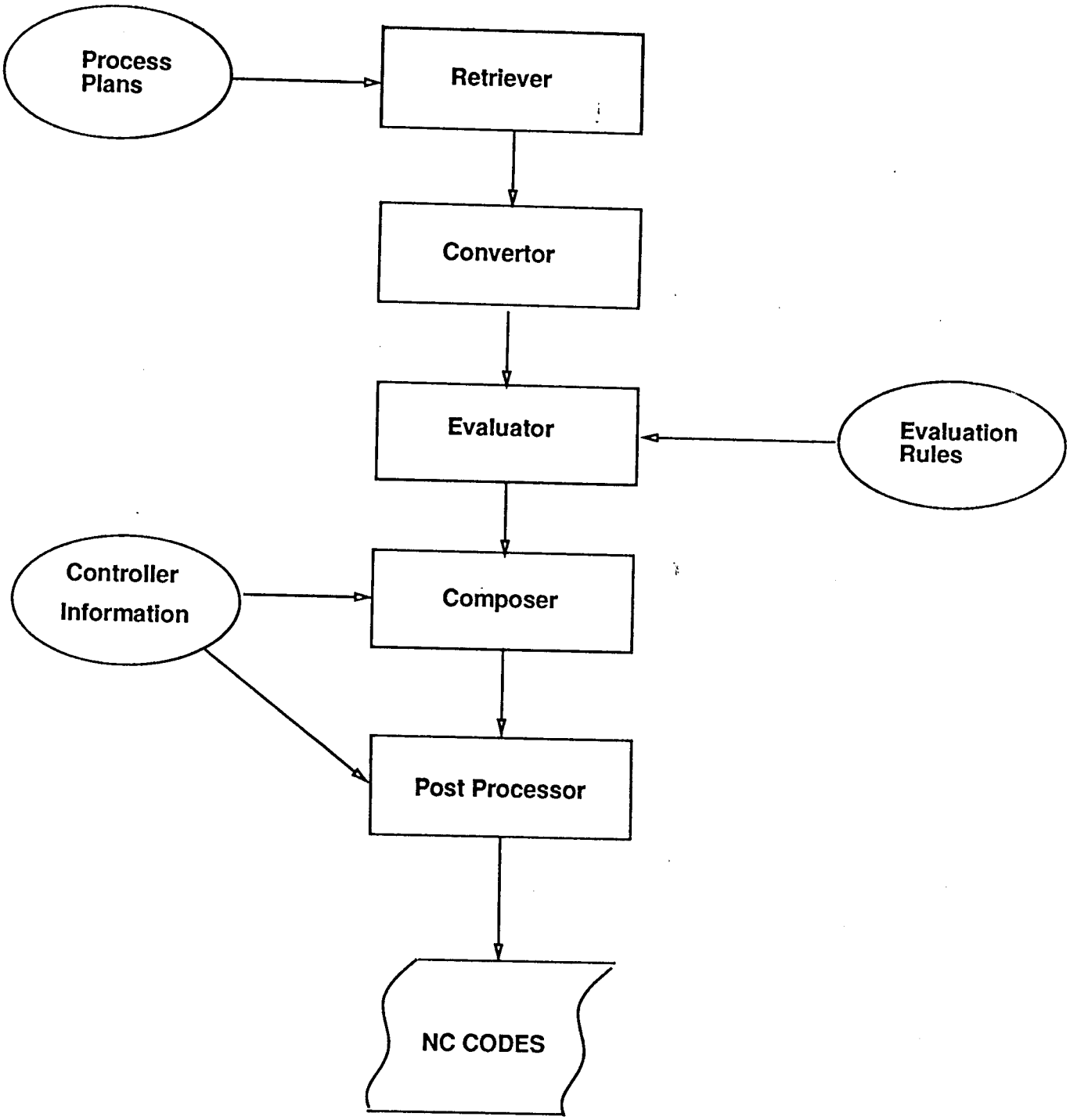


Figure 5: Schematic Representation Of The NC generator

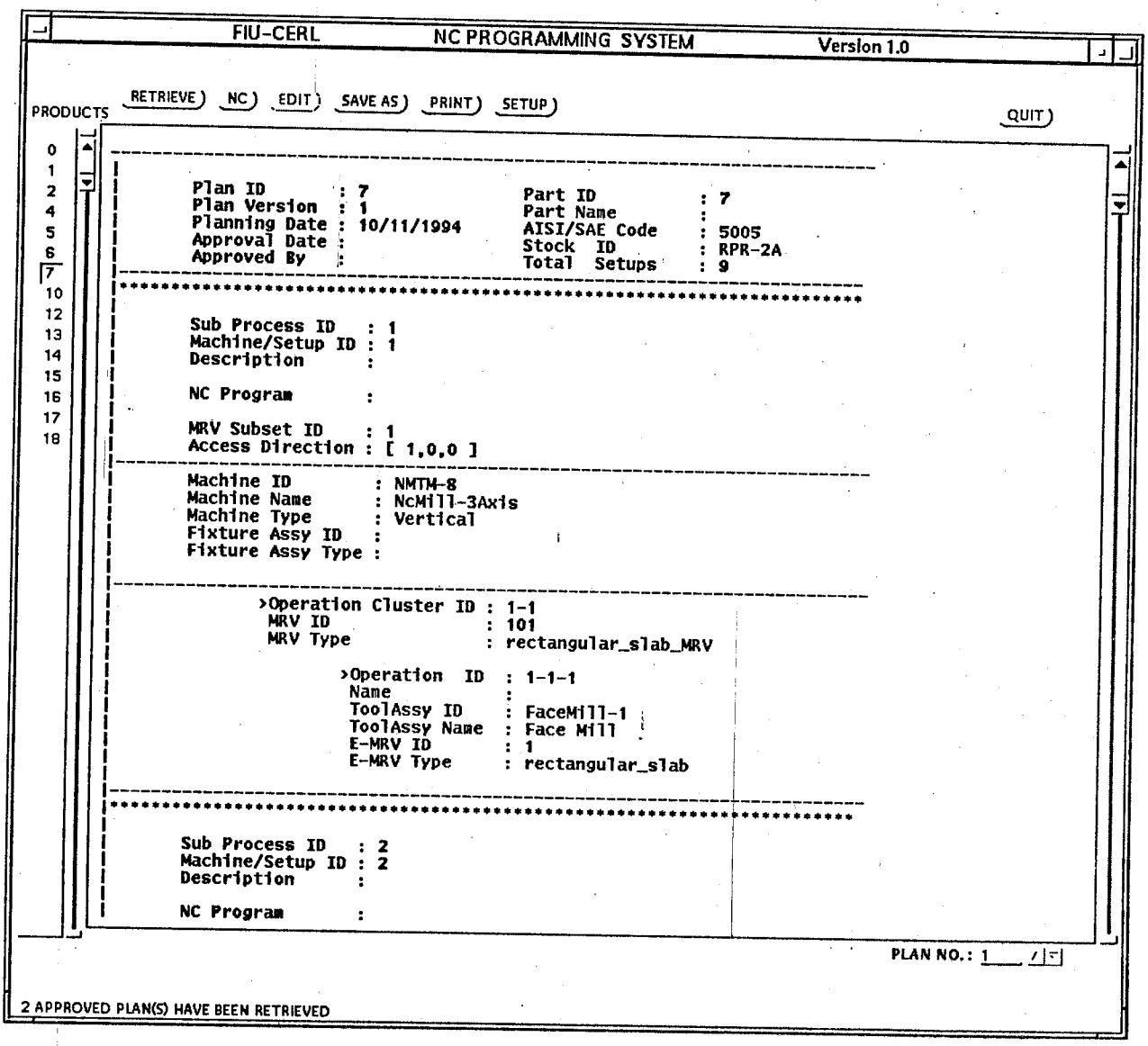


Figure 6: NC Program Generator Interface

- fixture data

4.1.1 Material Classes

In the Class RawMaterial, the attributes type, aisi-sae-code, bhnHardness, machinability, temper-code, length,shape,surfacefinish,density, and other are defined, which describe the basic properties of the material. Based on the RawMaterial, we have defined the classes CylindricalMaterial, PrismaticMaterial, Rectangle, Hexagon and Octagon to capture the different shape of the materials. Following is the class definitions for the Material Classes,

```
class   RawMaterial: public PVirtual {
  PString                  type;
  PString                  id;
  PString                  aisi_sae_code;
  int                      bhnHardness;
  float                    machinability;
  PString                  temper_code;

 protected:
  float                    length;

 private:
  PString                  shape;
  PString                  supplySource;
  PString                  specCertification;
  float                    weightPerFoot;
  PString                  location;
  PString                  status;
  float                    surfacefinish;

protected:
  float                   density;

public:
  RawMaterial();
  ~RawMaterial(); };
```

```
class   CylindricalMaterial: virtual public RawMaterial {
   float                    diameter;
 public:
   CylindricalMaterial();
   ~CylindricalMaterial(); };

class   PrismaticMaterial:virtual public RawMaterial {
 protected:
   float                    width;
 public:
   PrismaticMaterial();
   ~PrismaticMaterial(); };

class   Rectangle : public PrismaticMaterial {
   float                    thickness;
 public:
   Rectangle();
   ~Rectangle(); };

class   Hexagon: public  PrismaticMaterial {
   float                    flatDistance;
 public:
   Hexagon();
   ~Hexagon(); };

class   Octagon: public  PrismaticMaterial {
   float                    flatDistance;
 public:
   Octagon();
   ~Octagon(); };
```

4.1.2 Machine Classes

The Machine classes hierarchy has three level. The introduction of the middle level classes is to eliminate the data redundancy and reuse the class definition. At the root is the class Machine which describes the general attributes of machines. They include machineType, operationTypes, featureTypes, fixtureTypes, materialTypes, machineName, modelNo, serialNumber, machineId, tableWeightCap, maxTravelSpeed, throatDepth, workEnvX, workEnvY, workEnvZ, thName, thType, operatorsRequired, operatorClassification, machineRatePerHour and other attributes. The middle level includes a)Type which is the superclass of NCMill-3Axis, Mill-3Axis, Lathe, NCLathe and NCBed, b)Thermal which is the superclass of EDM, NCEDM and NCEDWC, c)Grind which is the superclass of NCJig, Jig and Surface, d)Saw which is the superclass of Band and Power, e)Drill which is the superclass of LightDuty, , Radial, Bed and NCBed, and f)Shape which does not have any subclass.

For the detailed information on the Machine Classes hierarchy, refer following class definition.

```
class   Machine: public PVirtual {
 PString                machineType;
 PString                operationTypes[30];
 PString                featureTypes[127];
 PString                fixtureTypes[30];
 PString                materialTypes[50];
 PString                companyName;
 PString                machineName;
 PString                modelNo;
 PString                serialNumber;
 PString                machineId;
 float                  tableWeightCap;
 float                  maxTravelSpeed;
 float                  throatDepth;
 float                  workEnvX;
 float                  workEnvY;
 float                  workEnvZ;
 PString                thName;
 PString                thType;
 int                    operatorsRequired;
 PString                operatorClassification[10];
```

```
    float                   machineRatePerHour;
    PString                 location;
    PString                 status;

    public:
    Machine();
    ~Machine(); };

class   Type:virtual public Machine {
    float                   spindleDiameter;
    float                   motorHp;
    int                     motorRpmMax;
    int                     motorRpmMin;
    PString                 spindlePosition;
    PString                 toolChangeModel;
    int                     toolChangeCap;
    PString                 toolChangeCalib;
  protected:
    float                   toolTravelX;
    float                   toolTravelZ;
    float                   positionAccuracy;
  private:
    float                   positionRepetability;
    PString                 coolantMediaTypes[5];
    PString                 coolantFlowTypes[5];
    PString                 machineCalibrationTypes[5];
    PString                 machineCalibrationDue;

  public:
    Type();
    ~Type(); };

class   Thermal:public Machine {
  protected:
    float                   toolTravelX;
    float                   toolTravelY;
```

```
    float                   positionAccuracy;
  private:
    float                   positionRepetability;
    float                   powerVolt;
    float                   powerAmp;
    PString                 coolant;
    float                   tSlotSize;
    float                   tSlotDistance;
    float                   boltHoleSize;
    float                   boltHoleDistance;
  public:
    Thermal();
    ~Thermal(); };

  class   EDM: public Thermal {
    PString                 toolChangeModel;
    int                     toolChangeCap;
    float                   toolTravelZ;

  public:
    EDM();
    ~EDM(); };

class   NCEDM:public Thermal {
    PString                 toolChangeModel;
    int                     toolModelCap;
    float                   toolTravelZ;
    PString                 ncControlType;
    PString                 mediaType[5];
    float                   memoryCapacity;
public:
    NCEDM();
    ~NCEDM(); };

class   NCMill_3Axis: public  Type {
    float                   memoryCapacity;
```

```
    PString                 ncControlType;
    PString                 mediaType[5];
    float                   toolTravelY;
    float                   toolChangeX;
    float                   toolChangeY;
    float                   toolChangeZ;
    float                   tSlotSize;
    float                   tSlotDistance;
    float                   boltHoleSize;
    float                   boltHoleDistance;
    float                   feedRateIPMMin;
    float                   feedRateIPMMax;
    PString                 specialEquip_Tool[20];

  public:
    NCMill_3Axis();
    ~NCMill_3Axis(); };

  class Mill_3Axis:public Type {
    float                   toolTravelY;
    float                   tSlotSize;
    float                   tSlotDistance;
    float                   boltHoleSize;
    float                   boltHoleDistance;
  public:
    Mill_3Axis();
    ~Mill_3Axis(); };

  class   Lathe:public  Type {
    PString                 headStkAttachment[10];
    PString                 tailStockType[10];

  public:
    Lathe();
    ~Lathe(); };
```

```
 class   NCLathe:public Type {
   PString                  ncControlType;
   PString                  mediaType[5];
   float                    memoryCapacity;
   PString                  headStkAttachment[10];
   PString                  tailStockType[10];

public:
   NCLathe();
   ~NCLathe(); };

 Class   NCEDWC: public Thermal {
   PString                  ncControlType;
   PString                  mediaType[5];
   float                    memoryCapacity;
   PString                  wireMaterial;
   float                    wireSizeMax;
   float                    wireSizeMin;
   float                    cuttingRate;
   float                    cutAngleMax;
public:
   NCEDWC();
   ~NCEDWC(); };

class   NCJig: public Grind {
   float                    peckStroke;
   float                    boltHoleSize;
   float                    boltHoleDistance;
   PString                  ncControlType;
   PString                  mediaType[5];
   float                    memoryCapacity;
   float                    positionAccuracy;
   float                    positionRepetability;
   PString                  coolantMediaTypes[5];
   PString                  coolantFlowTypes[5];
   PString                  machineCalibrationTypes[5];
```

```
    PString                 machineCalibrationDue;

  public:
    NCJig();
    ~NCJig(); };


  class   Jig:public Grind {
    float                   peckStroke;
    float                   boltHoleSize;
    float                   boltHoleDistance;
  public:
    Jig();
    ~Jig(); };



  class  Surface:public Grind {
    PString                 spindlePosition;
    PString                 chuckType;
   public:
    Surface();
    ~Surface(); };

  class   Saw: public Machine {
    float                   toolLength;
    PString                 toolPosition;
    float                   motorHp;
  protected:
    float                   toolTravelZ;
  public:
    Saw();
    ~Saw(); };

  class   Band:public  Saw {
    float                   toolSpeedMax;
```

```
    float                   toolSpeedMin;
  public:
    Band();
    ~Band(); };

class   Power:public Saw {
    float                   toolStroke;
    float                   toolFeedMax;
    float                   toolFeedMin;
  public:
    Power();
    ~Power(); };


class   Drill: virtual public Machine {
    float                   tSlotSize;
    float                   tSlotDistance;
    float                   toolDiameterMin;
    float                   toolDiameterMax;
  public:
    Drill();
    ~Drill(); };

class   LightDuty: public Drill {
    float                   spindleDiameter;
    float                   motorHp;
    float                   toolTravelZ;
    float                   motorRpmMin;
    float                   motorRpmMax;
  public:
    LightDuty();
    ~LightDuty(); };

class   Radial: public Drill {
    float                   spindleDiameter;
    float                   motorHp;
```

```
    float                   toolTravelZ;
    float                   toolAngleMax;
    float                   armRotationAngle;
    float                   motorRpmMin;
    float                   motorRpmMax;
  public:
    Radial();
    ~Radial(); };
class   Radial: public Drill {
    float                   spindleDiameter;
    float                   motorHp;
    float                   toolTravelZ;
    float                   toolAngleMax;
    float                   armRotationAngle;
    float                   motorRpmMin;
    float                   motorRpmMax;
  public:
    Radial();
    ~Radial(); };

class   Bed: public Drill {
    float                   spindleDiameter;
    float                   motorHp;
    float                   boltHoleSize;
    float                   boltHoleDistance;
    float                   toolTravelZ;
    float                   motorRPMMax;
    float                   motorRPMMin;
public:
    Bed();
    ~Bed(); };

class   NCBed: public Drill, public Type {
    PString                 ncControlType;
    PString                 mediaType[5];
    float                   memoryCapacity;
```

```
    float                   boltHoleSize;
    float                   boltHoleDistance;
    float                   toolTravelY;
  public:
    NCBed();
    ~NCBed(); };


  class   Shape: public Machine {
    float                   spindleDiameter;
    float                   motorHp;
    int                     motorRpm;
    PString                 toolPosition;
    float                   toolTravelX;
    float                   toolTravelY;
    float                   toolTravelZ;
    float                   positionAccuracy;
    float                   positionRepetability;
    float                   tSlotDistance;
    float                   tSlotSize;
    float                   boltHoleSize;
    float                   boltHoleDistance;
  public:
    Shape();
    ~Shape(); };
```

4.1.3 Tool Classes

The hierarchy of the tool classes is also three level. The introduction of the middle level classes is to eliminate the data redundancy and reuse the class definition. At the root is the class ToolAssembly which describes the general attributes of tool assemblies. They include assemblyName, assemblyId, cbSource, thType, thName, machineType, operationType, lengthMax, featureType, cbName, cbMaterial, cbGrade, materialCutCapability , surfaceFinishCapability and so on. At the middle level there are Mill, Lathe, Thermal, Saw, Shape and Drill, which abstract the low level class definitions.

For the detailed information on the tool assembly hierarchy, refer following class definition.

```
class   ToolAssembly: public PVirtual {
  PString                  assemblyName;
  PString                  assemblyId;
  PString                  cbSource;
  PString                  thType;
  PString                  thName;
  PString                  machineType;
  PString                  operationType[30];
  float                    lengthMax;
  PString                  featureType[127];
  PString                  cbName;
  PString                  cbMaterial;
  PString                  cbGrade;
  PString                  materialCutCapability[20];
  float                    surfaceFinishCapability;
  int                      quantity;
  PString                  location;
  PString                  status; };

class   MillTA:public ToolAssembly {
  float                    cblength;
  float                    cbDiameter;
  float                    cbWidth;
  float                    diameterMax;
  float                    widthMax;
  int                      edgeNumber;
  float                    edgeRadius;
  float                    maxDepthofcut;
  float                    toolLife;
  float                    accumulateLife; };


class   FaceTA:public MillTA{
```

```
    PString                 insertType[10];
    float                   insertCapacity;
    PString                 iGrade;
    float                   iIc;
    PString                 iCatalogCode;
    PString                 RoughFinish; };

  class   MDrill:public MillTA{

    PString                 insertType[10];
    float                   insertCapacity;
    PString                 iGrade;
    float                   iIc;
    PString                 iCatalogCode;
    float                   pointAngle;
    PString                 RoughFinish; };

class   Side:public MillTA{

    PString                 insertType[10];
    float                   insertCapacity;
    PString                 iGrade;
    float                   iIc;
    PString                 iCatalogCode;
    int                     cuttingSides;
    PString                 RoughFinish; };

class   MBore:public MillTA{
    float                   cbDiameterMin;
    float                   cbDiameterMax;
    PString                 RoughFinish; };

class   MCenter:public MillTA{
```

```
    PString                 cbType;
    PString                 sizeDesignation; };

class   End:public MillTA{

    PString                 insertType[10];
    float                   insertCapacity;
    PString                 iGrade;
    float                   iIc;
    PString                 iCatalogCode;
    float                   cbAngle;
    PString                 RoughFinish;
    PString                 centerCutCapab; };

class   MReam:public MillTA{ };

class   Peripheral:public MillTA{

    PString                 insertType[10];
    float                   insertCapacity;
    PString                 iGrade;
    float                   iIc;
    PString                 iCatalogCode; };


class   LatheTA: public ToolAssembly {
    float                   cbLength;
    float                   edgeRadius;
    float                   toolLife;
    float                   accumulateLife; };

class   SinglePoint: public LatheTA {
    float                   cbWidth;
    float                   cbThickness;
    float                   widthMax;
```

```
    float                   thicknessMax;
    PString                 insertType[10];
    float                   insertCapacity;
    PString                 iGrade;
    float                   iIc;
    PString                 iCatalogCod };

class   External:public SinglePoint { };

class   Internal:public SinglePoint { };

class   Groove:public   LatheTA {
   float                    cbWidth;
   float                    cbThickness;
   float                    widthMax;
   float                    thicknessMax; };

class   LDrill:public   LatheTA {
   float                    cbDiameter;
   float                    diameterMax;
   PString                  insertType[10];
   float                    insertCapacity;
   PString                  iGrade;
   float                    iIc;
   PString                  iCatalogCode;
   float                    maxDepthofcut;
   PString                  RoughFinish; };

class   Knurl:public    LatheTA {
   PString                  cbForm;
   float                    cbWidth;
   float                    cbThickness;
   float                    widthMax;
   float                    thicknessMax; };

class   LBore:public    LatheTA {
```

```
    float                   cbDiameter;
    float                   diameterMax;
    PString                 RoughFinish; };

class   Form:public    LatheTA {
    PString                 cbForm;
    float                   cbWidth;
    float                   cbThickness;
    float                   widthMax;
    float                   thicknessMax; };

class   LReam:public    LatheTA {
    float                   cbDiameter;
    float                   diameterMax; };

class   LCenter:public LatheTA{
    PString                 cbType;
    PString                 sizeDesignation;
    float                   cbDiameter;
    float                   maxDepthofcut; };

class   ThermalTA:public ToolAssembly {
    float                   cbDiameter; };

class   Wire: public ThermalTA { };

class   GrindTA:public ToolAssembly {
    float                   cbDiameter;
    float                   cbWidth;
    float                   diameterMax;
    float                   widthMax;
    PString                 abrasive;
    int                     grainSize;
    PString                 bond;
    char                    grade;
    int                     stucture; };
```

```
class   SawTA: public ToolAssembly {
   float                    cblength;
   float                    cbWidth;
   float                    cbThickness;
   int                      toothPitch;
   PString                  toothSet;
   PString                  toothForm; };

class   BandTA : public SawTA { };

class   Blade : public SawTA { };

class      ShapeTA: public ToolAssembly {
   float                    cblength;
   float                    cbWidth;
   float                    cbThickness;
   float                    lengthMax;
   float                    widthMax; };

class   DrillTA: public ToolAssembly {
   float                    cbLength;
   float                    cbDiameter;
   float                    edgeRadius;
   float                    diameterMax;
   int                      edgeNumber;
   float                    maxDepthofcut;
   float                    toolLife;
   float                    accumulateLife; };

class  DDrill:public DrillTA {

   PString                  insertType[10];
   float                    insertCapacity;
   PString                  iGrade;
   float                    iIc;
```

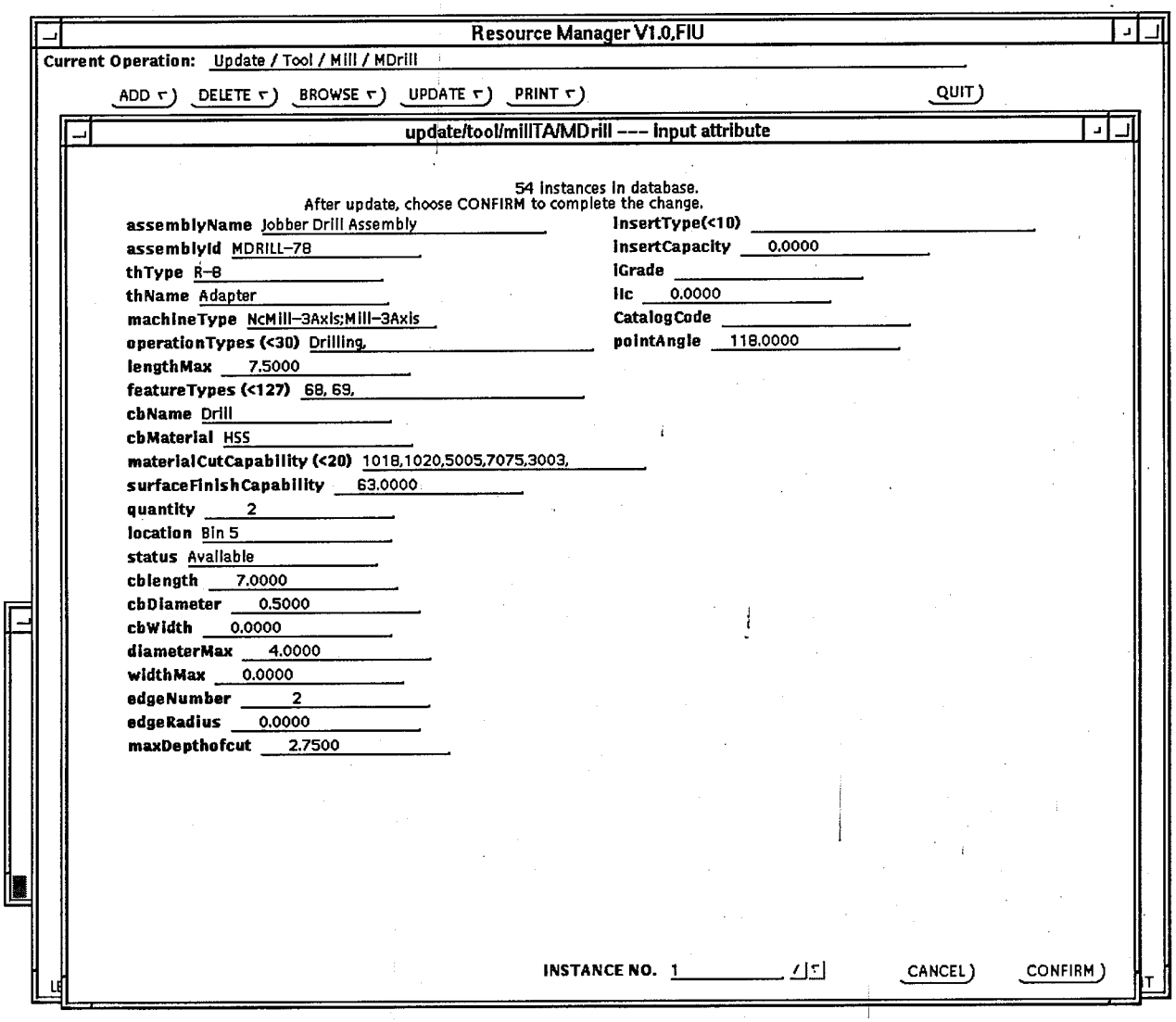


Figure 7: Database Manager Interface

```
    PString                 iCatalogCode;
    float                   pointAngle;
    PString                 RoughFinish; };

class  DReam:public DrillTA { };

class  DBore:public DrillTA {

    float                   cbDiameterMin;
    float                   cbDiameterMax;
    PString                 RoughFinish; };

class   DCenter:public DrillTA{
    PString                  cbType;
    PString                  sizeDesignation; };
```

4.2 The Database Manager

We have developed a comprehensive integrated database manager using XVIEW GUI. By using this user-friendly interface, a user can browse/update/add/delete/print all machine, tool or material data stored in the database, See Figure 7. We have designed a hierarchical memu structure which reflects the class definitions. When the users use the interface to manage the manufacturing resource data, the only choice they neeed to do is to click one of five buttons located on the top of the main window. Those five buttons are ADD, BBOWSE, UPDATE, DELETE and PRINT. After the users selected a button , the system will display a menu to guide the users to choose the correct menu item.

5 Manufacturing Knowledge Base and Its Manager

Knowledge is the symbolic representation of aspects of some name d universe of discourse. It is the key element of decision-level automation and the competitive edge for manufacturing system. In our system we have systemized the knowledge in the field of manufacturing.They include:

1. rules for form feature recognition;

2. rules for operation selection;

3. rules for machine/tool selection;

4. rules for machining parameter selection.

5.1 Knowledge Represention Schemes

Representation of knowledge is a combination of data structures and interpretive procedures. In the system we have used procedural representation, production system, object-oriented frame, and integrated the knowledge represented in the difference approaches in the framework of object-oriented methodology.

5.1.1 Rules For Form Feature Recognition

Two formats of feature recognition rules are developed. One format is for cuboid and wedge solids whose bounding faces are known before they are instantialized. The other is for solids whose bounding faces are determined when they are constructed. The first format of feature recognition rules takes the following form:

IF solid-type & face-pattern & boolean-op THEN feature-type

where solid-type is either a cuboid or a wedge; face-pattern is a subset of faces bounding a cuboid or a wedge, boolean-op is either a Boolean addition or subtraction. For example, a feature rule that will recognize the feature type as blind-step can be written as:

IF "Cuboid" & 1, 4, 5 & "-" THEN blind-step

The second format of rules look like:

IF solid-type & top-face & bottom-face & face-ratio & bool-op THEN feature-type

where solid-type and bool-op have the same meaning as defined in the first format. Top-face and bottom-face take a binary value of 1 or 0, indicating the existence of the top face and the bottom face on the product. For instance, following is a rule in the second format.

IF "Cylinder" & 0 & 9 & 1.0 & "-" THEN cylindrical-through-hole

5.1.2 rules for operation selection

In the system , we have defined a table called FMT(Feature-Machine-Tool), which describes the possibility of the machine/tool combination for production of the given feature. For example the following is the FMT table for CYLINDRICAL-THRU-STEP,

From the table entry, we can see that following machine/tool combination can be used to produce a CYLINDRICAL-THRU-STEP:

MRV TYPE: CYLINDRICAL_THRU_STEP

MRV ID: 64

TOOL ASSEMBLIES	Number		MACHINES 1 Mill-3-Axis	2 NcMill-3Axis	3 Lathe-2Axis	4 NcLathe-2Axis	5 EDM	6 NcEDM	7 NcEDWC	8 SurfaceGrind	9 JigGrind	10 NcJigGrind	11 BandSaw	12 PowerHackSaw	13 Shape	14 LightDutyDrill	15 BedDrill	16 NcBedDrill	17 RadialDrill
Mill	1	Face																	
	2	Side																	
	3	End	X	X															
	4	Peripheral	X	X															
	5	MCenter																	
	6	MDrill																	
	7	MReam																	
	8	MBore	X	X															
Lathe	9	ExternalSP																	
	10	InternalSP			X	X													
	11	Form																	
	12	Knurl																	
	13	Groove																	
	14	LCenter																	
	15	LDrill																	
	16	LReam																	
	17	LBore			X	X													
Thermal	18	Electrode					X	X											
	19	Wire							X										
Grind	20	Grind								X	X	X							
Saw	21	Band																	
	22	Blade											X						
Shape	23	Shape																	
Drill	24	DCenter																	
	25	DDrill																	
	26	DReam																	
	27	DBore																	

Figure 8: FMT/M/T Table for Cylindrical Through Step

- (Mill-3Axis, End)
- (NcMill-3Axis, End)
- (Mill-3Axis, Peripheral)
- (NcMill-3Axis, Peripheral)
- (Mill-3Axis, MBore)
- (NcMill-3Axis, MBore)
- (Lathe-2Axis, InternalSP)
- (NcLathe-2Axis, InternalSP)
- (Lathe-2Axis, LBore)
- (NcLathe-2Axis, LBore)
- (EDM, Electrode)
- (NcEDM, Electrode)
- (NcEDWC, Wire)
- (SurfaceGrind, Grind)
- (JigGrind, Grind)
- (NcJigGrind, Grind)
- (BandSaw, Band)

In the FMT table , we also have defined the cost and flexablitiy of the machine/tool combination. For example, the cost and flexablity of the first combination is 3, 2 respectively, the last one's are 1, 15. So using BandSaw/Band to finish the feature is much more cheap than than using Mill-3Axis/End, but the latter is more flexible. It notes that the rule to select the machine and tool assembly instances for the given MRV dimension is linked to the FMT table entry. We call the tuple (machine, tool, cost, flexibility, link torule) the Combo. Following is the definition for Class Combo.

```
class Combo: public PObject {

     PString assembly;
     PString machine;
    public:

     int cost_rank;
     int mach_flex_rank;

     Link<Rule>  rule;

     int toolassembly_index();
     int machine_index();
     void set_cost_rank(int cost);
     void set_mach_flex_rank(int flex);
     PString toolassembly_name();
     PString machine_name(); };
```

5.1.3 rules for machine/tool selection

The rules for machine/tool selection are important part of our knowledge base, They play a key role in the process planning. Following is a sample of rules in procedure representation, which is used for CYLINDRICAL-THROUGH-HOLE with Mill-3Axist/MDrill or NCMill-3Axis/MDrill :

```
/*************************************************************
*
*
* rule for cylindrical_through_hole Mill_3Axis/MDrill
*                                   NCMill_3Axis/MDrill
*
*************************************************************/
int rule_1( int flag, double d1, double d2, double height,
           char *machine_name, Type *machine, FinalMRV *aMRV)
{
```

```
float testlengthMax;
char  *testthname;
char  *testthtype;
char  *testmaterialCutCapability;

RawMaterial *material = aMinStock;


if(!strcasecmp(machine->get_spindlePosition(), "Horizontal"))
  testlengthMax = machine->get_workEnvY();
else
 testlengthMax=machine->get_workEnvZ()-material->get_max_dimension();

 testthname = new char[strlen(machine->get_thName())+1];
 assert(testthname !=0);
 strcpy(testthname, machine->get_thName());


 testthtype = new char[strlen(machine->get_thType())+1];
 assert(testthtype !=0);
 strcpy(testthtype, machine->get_thType());


 testmaterialCutCapability =
         new char[strlen(material->get_aisi_sae_code())+1];
 assert(testmaterialCutCapability !=0);
 strcpy(testmaterialCutCapability, material->get_aisi_sae_code());

// find a instance of MDrill Toolassembly

LinkVstr<MDrill> mdrill_toolassemblies = MDrill::find_all();

LinkVstr<MDrill> selected_mdrill;

if (mdrill_toolassemblies.size()<=0)
```

```
 {
 return NO_TOOLASSEMBLY_INSTANCE;
 };


 int j, selectedmdrill = -1;
 MDrill *mdrill;

 for(j=0;j<mdrill_toolassemblies.size();j++)
 {
  mdrill = mdrill_toolassemblies[j];

  if ( In_matCutCapability(testmaterialCutCapability,
                           mdrill->get_materialCutCapability()) &&
      !strcasecmp(mdrill->get_thName(), testthname) &&
      !strcasecmp(mdrill->get_thType(), testthtype) &&
      (mdrill->get_cbDiameter() >= d1) &&
      (mdrill->get_cbDiameter() <= d2) )
  {
     if ( (mdrill->get_lengthMax() <= testlengthMax) &&
          (mdrill->get_cblength() >=  height)  )
      {
          selectedmdrill = j;
          break;
      };
  }
 }

 if (selectedmdrill == -1)
 {
  cout<<"############  TOOL SELECTION FAILS ################"
     <<endl<<endl;
  return -2;
 }
 else
 {
```

```
selected_mdrill.add(mdrill_toolassemblies[selectedmdrill]);

cout<<endl<< "*** SELECTED MDRILL IS "
    <<mdrill_toolassemblies[selectedmdrill]->get_assemblyId()
    <<endl<<endl;

 float cb = mdrill_toolassemblies[selectedmdrill]->get_cbDiameter();
 if (cb > 0.5 )
 {
  cout<<"call the module itself again!"<<endl;
  if(!rule_1(1, cb-0.3, cb-0.25,height,  machine_name,
             machine,  aMRV))
  {
    tool_types[tool_ids_index] =
       mdrill_toolassemblies[selectedmdrill]->get_assemblyName();
    tool_ids[tool_ids_index++] =
      mdrill_toolassemblies[selectedmdrill]->get_assemblyId(
 );

    Element_MRV *e_mrv = new Persistent
                    Element_MRV(0, "Tube_Elemental_MRV");
    e_mrv->set_dimension(
       mdrill_toolassemblies[selectedmdrill]->get_cbDiameter(),
       mdrill_toolassemblies[selectedmdrill]->get_cbDiameter(),
       height);

aMRV->get_location(tmp);
e_mrv->set_location(tmp[0], tmp[1], tmp[2]);
aMRV->get_orientation(tmp);
e_mrv->set_orientation(tmp[0], tmp[1], tmp[2]);

  if(flag == 0)
  {
    e_mrv->set_rough_finish("Finish");
    e_mrv->set_allowance(0.0);
  }
```

```
    else
    {
      e_mrv->set_rough_finish("Rough");
      e_mrv->set_allowance(
          lookup_allowance("MDrill", aMinStock->get_aisi_sae_code()));
   }

    E_mrvs.add(e_mrv);

    return 0;

   }
   else
    return -1;
 }
 else
 {
 cout<<"call centermdirll_mrv rules"<<endl;
if(!rule_0(1, cb,cb,  machine_name, machine,  aMRV))
  {

   tool_types[tool_ids_index] =
      mdrill_toolassemblies[selectedmdrill]->get_assemblyName();
   tool_ids[tool_ids_index++] =
      mdrill_toolassemblies[selectedmdrill]->get_assemblyId();

   Element_MRV *e_mrv = new Persistent
                    Element_MRV(0, "Tube_Elemental_MRV");
   e_mrv->set_dimension(
       mdrill_toolassemblies[selectedmdrill]->get_cbDiameter(),
       mdrill_toolassemblies[selectedmdrill]->get_cbDiameter(),
       height);

aMRV->get_location(tmp);
e_mrv->set_location(tmp[0], tmp[1], tmp[2]);
aMRV->get_orientation(tmp);
```

```
    e_mrv->set_orientation(tmp[0], tmp[1], tmp[2]);

     if(flag == 0)
     {
      e_mrv->set_rough_finish("Finish");
      e_mrv->set_allowance(0.0);
     }
     else
     {
      e_mrv->set_rough_finish("Rough");
      e_mrv->set_allowance(
            lookup_allowance("MDrill", aMinStock->get_aisi_sae_code()));
     }

     E_mrvs.add(e_mrv);

    return 0;

    }
    else
     return -1;

  }

};

return 0;
}
```

We have implemented following rule sets in our system:

1. rule for centerdrill-mrv with Mill-3Axis/Mcenter or NCMill-3Axis/Mcenter
2. rule for centerdrill-mrv with Lighdutydrill/Dcente
3. rule for cylindrical-through-hole Mill-3Axis/MDrill or NCMill-3Axis/MDrill

4. rule for cylindrical-through-hole with lightduty/DDrill
5. rule for cylindrical-through-hole Mill-3Axis/MBore or NCMill-3Axis/MBore
6. rule for cylindrical-through-hole with lightduty/DBore
7. rule for cylindrical-through-hole Mill-3Axis/MReam or NCMill-3Axis/MReam
8. rule for cylindrical-through-hole with lightduty/DReam
9. rule for cylindrical-through-hole with MILL-3Axis/End or NCMill-3Axis/End
10. rule for cylindrical-blind-hole with MILL-3Axis/End or NCMill-3Axis/End
11. rule for cylindrical-blind-hole with lightduty/DDrill
12. rule for cylindrical-blind-hole Mill-3Axis/MBore or NCMill-3Axis/MBore
13. rule for cylindrical-blind-hole Mill-3Axis/MReam or NCMill-3Axis/MReam
14. rule for cylindrical-blind-hole with lightduty/DReam
15. rule for cylindrical-blind-hole Mill-3Axis/MDrill or NCMill-3Axis/MDrill
16. rule for rounded-cuboid-pocket with MILL-3Axis/End or NCMill-3Axis/End
17. rule for rounded-diamond-pocket with MILL-3Axis/End or NCMill-3Axis/End
18. rule for rounded-traperoid-pocket with MILL-3Axis/End or NCMill-3Axis/End
19. rule for regular-cuboid-pocket with EDM/Electrode
20. rule for rounded-cuboid-blind-slot with MILL-3Axis/End or NCMill-3Axis/End
21. rule for regular-cuboid-blind-slot with EDM/Electrode
22. rule for regular-cuboid-thru-slot with MILL-3Axis/End or NCMill-3Axis/End
23. rule for regular-cuboid-thru-slot with MILL-3Axis/Side or NCMill-3Axis/Side
24. rule for regular-cuboid-thru-slot with SurfaceGrind/Grind
25. rule for regular-cuboid-thru-step with MILL-3Axis/Side or NCMill-3Axis/Side
26. rule for regular-cuboid-thru-step with MILL-3Axis/End or NCMill-3Axis/End

27. rule for rectangular-slab with MILL-3Axis/Face or NCMill-3Axis/Face

28. rule for rectangular-slab with MILL-3Axis/End or NCMill-3Axis/End

29. rule for rectangular-slab with MILL-3Axis/Peripheral or NCMill-3Axis/Peripheral

30. rule for rectangular-slab with BandSaw/Saw

5.1.4 rules for machining parameter selection

The rules for machining parameter selection are in the tabular format. Given the material type, the cut body diameter of the tool assembly, the provided query procedure will retrieve the machining parameters for the caller. For example, Figure 9 is speed and feed parameter selection table for the end milling operation.

In the knowledge base we have implemented the following tables:

1. End Milling Parameter Table

2. Boring Parameter Table

3. Reaming Parameter Table

4. Drilling Parameter Table

5. Face Milling Parameter Table

5.2 The Knowledge Manager

We have developed a knowledge base manager with graphical user interface using XVIEW GUI(See Figure 10). On the background of the main window is displayed a FMT which name is near the message item FMT NAME. You can change the currently displayed FMT by clicking the button $->$. By clicking the mouse left on the table cell you can add or delete a combo on the current FMT.

End Milling Speeds and Feeds

	Hard	Hard				ipt (given cbDiameter <=)			
Material	Low	High	DOC(R)	cbMaterial	fpm	0.375	0.5	0.75	1
1018	85	124	0.02	HSS	190	0.001	0.002	0.004	0.005
1018	85	124	0.06	HSS	145	0.002	0.003	0.005	0.006
1018	85	124	-4	HSS	130	0.001	0.002	0.004	0.005
1018	85	124	0.02	Carbide	550	0.0015	0.0035	0.006	0.007
1018	85	124	0.06	Carbide	425	0.0025	0.004	0.006	0.008
1018	85	124	-4	Carbide	365	0.002	0.003	0.005	0.006
1018	125	174	0.02	HSS	210	0.001	0.002	0.004	0.005
1018	125	174	0.06	HSS	160	0.002	0.003	0.005	0.006
1018	125	174	-4	HSS	140	0.001	0.002	0.004	0.005
1018	125	174	0.02	Carbide	520	0.0015	0.0035	0.006	0.007
1018	125	174	0.06	Carbide	400	0.0025	0.004	0.006	0.008
1018	125	174	-4	Carbide	350	0.002	0.003	0.005	0.006
1018	175	224	0.02	HSS	170	0.001	0.002	0.003	0.004
1018	175	224	0.06	HSS	125	0.002	0.003	0.004	0.005
1018	175	224	-4	HSS	110	0.001	0.002	0.003	0.004
1018	175	224	0.02	Carbide	485	0.0015	0.003	0.005	0.006
1018	175	224	0.06	Carbide	370	0.0025	0.004	0.006	0.007
1018	175	224	-4	Carbide	325	0.002	0.003	0.005	0.006
1018	225	275	0.02	HSS	120	0.001	0.002	0.003	0.004
1018	225	275	0.06	HSS	90	0.002	0.003	0.004	0.005
1018	225	275	-4	HSS	80	0.001	0.002	0.003	0.004
1018	225	275	0.02	Carbide	455	0.001	0.002	0.004	0.005
1018	225	275	0.06	Carbide	350	0.002	0.003	0.005	0.007
1018	225	275	-4	Carbide	300	0.0015	0.0025	0.004	0.005
1020	85	124	0.02	HSS	220	0.001	0.002	0.004	0.005
1020	85	124	0.06	HSS	170	0.002	0.003	0.005	0.006
1020	85	124	-4	HSS	150	0.001	0.002	0.004	0.005
1020	85	124	0.02	Carbide	800	0.0015	0.0035	0.006	0.007
1020	85	124	0.06	Carbide	625	0.0025	0.004	0.006	0.008
1020	85	124	-4	Carbide	490	0.002	0.003	0.005	0.006
1020	125	174	0.02	HSS	210	0.001	0.002	0.004	0.005
1020	125	174	0.06	HSS	160	0.002	0.003	0.005	0.006
1020	125	174	-4	HSS	140	0.001	0.002	0.004	0.005
1020	125	174	0.02	Carbide	725	0.0015	0.0035	0.006	0.007
1020	125	174	0.06	Carbide	560	0.0025	0.004	0.006	0.008
1020	125	174	-4	Carbide	435	0.002	0.003	0.005	0.006

Figure 9: End Milling Machining Parameter Table

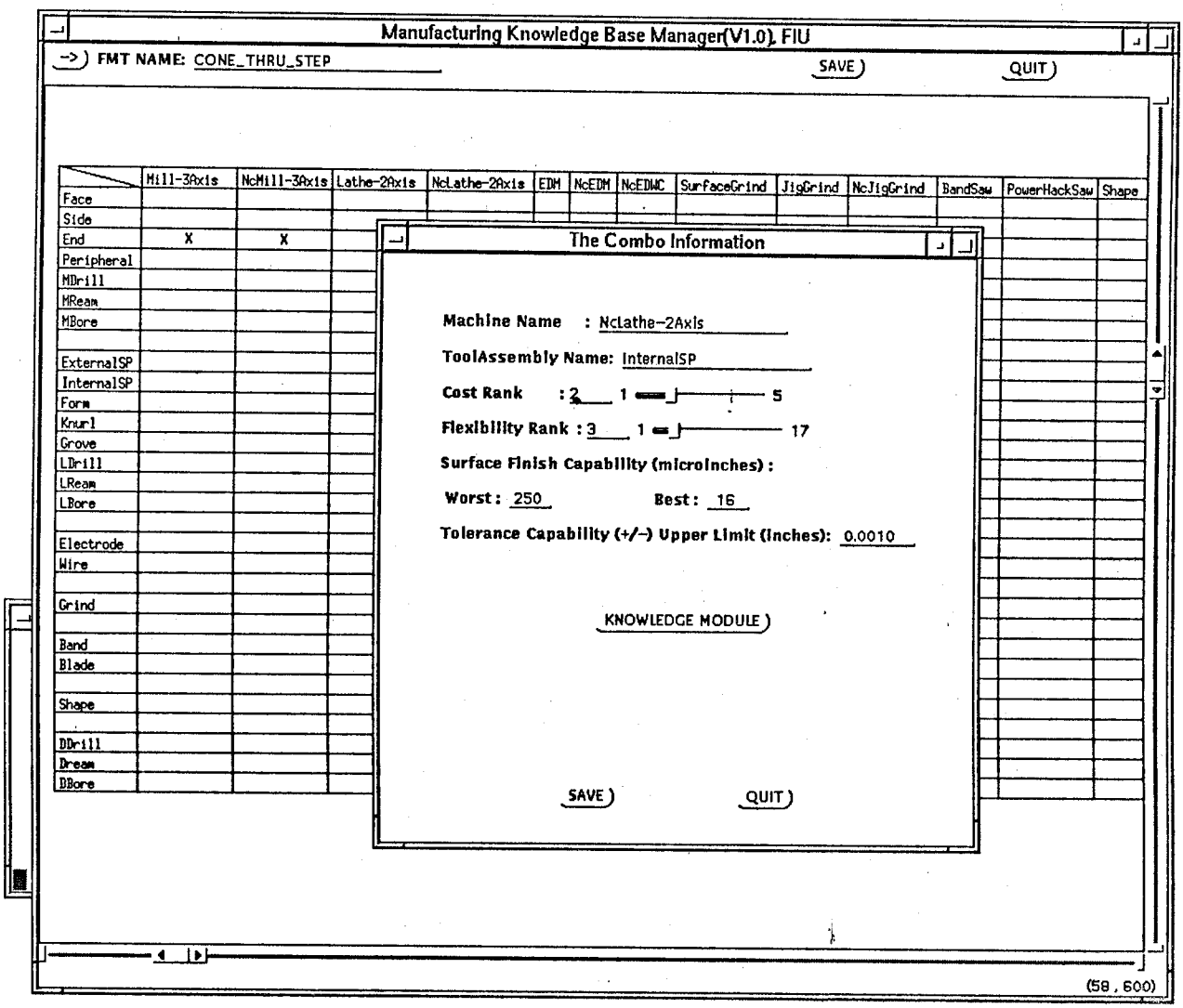


Figure 10: Knowledge Base Manager Interface

Appendix H: List of Related Publications

a. Journal papers

1. Chen, C., and Sanchez, M., "*Machining Features Extraction from a Feature-based Product Model*," Journal of Intelligent Manufacturing, in review process.

2. Wu, J., and Chen, C., "*Object-oriented Implementation of a Feature Modeling System for CAD/CAM Integration*," Computer Aided Design, in review process.

3. Chen, C., and Sagarsee, S., "*A Feature-based Approach to Producibility Evaluation of Machined Component Designs*," International Journal of Computer Integrated Manufacturing, in review process.

4. Chen, C., and Wu, J., "*A Hybrid Feature Modeling Scheme Supporting Feature Deletion/Undoing and Extraction*," Computer and Graphics, in review process.

5. Chen, C., and Wu, J., "*Integrated Surface and Volume Feature Modeling Approach to CAD/CAM Systems Integration*," International Journal of Integrated Manufacturing Systems, accepted for publication.

6. Chen, C., Swift, F., Lee, S., Ege, R., and Shen, Q., "*Development of a Feature-based and Object-Oriented Concurrent Engineering System*," Journal of Intelligent Manufacturing, 1994, Vol.5, pp.23-31.

b. Book chapters

1. Chen, C., Chow, J., Swift, F., and Lee, S., "*A Concurrent Design Support System in Concurrent Engineering*," Concurrent Engineering, by P. Gu and A. Kusiak (eds), Elsevier Science Publisher (North Holland), 1993, pp.235-245.

2. Chen, C., and Wu, J., "*Product Modeling and Exchange*," Concurrent Engineering, by P. Gu and A. Kusiak (eds.), Elsevier Science Publisher (North Holland), 1993, pp.299-313.

c. Refereed conference proceedings

1. Chen, C., "*A Framework for Concurrent Process Planning*," Proceedings of the Fourth International Conference on Flexible Automation and Information Management, Blacksburg, VA., May 9-12, 1994.

2. Chen, C., and Wu, J., "*A Hybrid CSG/Brep Representation Scheme for Feature Modeling*," Proceedings of CSG'94 Set-Theoretic Solid Modeling: Techniques and Applications Conference, Winchester, U.K., April, 1994, PP.291-303.